David Pedrosa Branco

MIAUDIO – Matriz de Mistura de Áudio

David Pedrosa Branco

MIAUDIO – Matriz de Mistura de Áudio

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Dr. José Neto Vieira, Professor do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

o júri / the jury

presidente / president

António Manuel de Brito Ferrari Almeida
Professor Catedrático da Universidade de Aveiro

vogais / examiners committee

José Manuel Neto Vieira
Professor Auxiliar da Universidade de Aveiro (orientador)

Iouliia Skliarova
Professora Auxiliar da Universidade de Aveiro (co-orientadora)

António José Duarte Araújo
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto

agradecimentos / acknowledgements

Gostaria de agradecer ao professor José Neto Vieira e professora Iouliia Skliarova por todos os ensinamentos e conselhos que me deram durante este último ano. O método de trabalho aplicado e a organização proporcionada foram factores cruciais para a conclusão deste projecto.
Aproveito também para agradecer ao grupo da sala 317 pelo excelente ambiente de trabalho criado e pela ajuda nos momentos mais desafiantes.
Por último, um agradecimento à minha família que me acompanhou e apoiou durante todo o processo.

Resumo

Nos dias de hoje, a música electroacústica recorre cada vez mais a técnicas de difusão de som. Sistemas multicanal como o BEAST e o SARC foram construídos para que o utilizador consiga controlar independentemente a intensidade de cada canal de entrada por cada uma das saídas, permitindo assim, a criação de diferentes cenários sonoros, logo, a movimentação e espacialização do som. O sistema aqui desenvolvido (MIAUDIO) é um sistema de tempo real que permite a mistura de 8 canais de entrada analógicos por 32 saídas, independentemente. Foi adoptada uma solução por hardware sendo que a mistura é realizada numa Field Programmable Gate Array (FPGA). Os sinais de entrada são tratados, convertidos para digital e processados na FPGA. Os parâmetros da mistura são enviados pelo computador para a FPGA via USB. Assim sendo, o utilizador controla a intensidade dos 8 canais de entrada nas 32 saídas, independentemente, possibilitando a movimentação do som. MIAUDIO foi implementado com sucesso. Esta solução é uma solução low-cost (<500€), cujo tempo de desenvolvimento foi relativamente curto. Outro aspecto diferenciador em MIAUDIO relaciona-se com o facto do computador que define a mistura ter muito poucos recursos alocados uma vez que a mistura é feita em hardware. Todos os canais foram testados com recurso ao equipamento Precision One tendo sido obtidos bons resultados.

Abstract

Electroacoustic music is turning more and more to the sound diffusion techniques. Multichannel sound systems like BEAST and SARC are built so that the musician can independently control the intensity of several audio channels. This feature provides the possibility of creating several sound diffusion scenarios, i.e., immersion and the possibility of movement around the audience. The developed system (MIAUDIO) is a real-time sound diffusion system currently able to mix up to 8 audio input channels through 32 outputs channels. A hardware solution was adopted using a Field Programmable Gate Array (FPGA) to perform the mixture. The analogue audio signals are conditioned, converted to digital format by several analogue-to-digital converters and then sent to the FPGA that is responsible to perform the mixing algorithm. The host computer connects to the FPGA via USB and is responsible for supplying the parameters that define the audio mixture. Being so, the user has control over the input levels through the output channels independently so that sound movement is possible. MIAUDIO was successfully implemented with a low-cost solution (<500€) when compared with similar systems. The developing time was relatively short and resources of the host computer operating system are almost not occupied because the mixture is done completely in FPGA. All the channels were tested using Precision One system with very good results.

Índice

Índice de Tabelas

Índice de Figuras

Lista de Acrónimos

ADC	Analog-to-Digital Converter
BEAST	Birmingham ElectroAcoustic Sound Theater
CLB	Configurable Logic Block
DECA	Departamento de Comunicação e Arte
DPCUTIL	Digilent Port Communications Utility
DCM	Digital Clock Manager
DSP	Digital Signal Processor
DAC	Digital-to-Analog Converter
DLL	Dynamic Link Library
FPGA	Field Programmable Gate Array
FIFO	First-In-First-Out
IOB	Input/Output Block
LED	Light-Emitting Diode
LUT	Look-Up Table
MSP	Max Signal Processing
MIDI	Musical Instrument Digital Interface
PCI	Peripheral Component Interconnect
PC	Personal Computer
PCB	Printed Circuit Board
RAM	Random Access Memory
SARC	Sonic Arts Research Center
USB	Universal Serial Bus

Capítulo 1 – INTRODUÇÃO

1.1 – Enquadramento

O avanço da tecnologia tem proporcionado diversas alterações no nosso quotidiano. Áreas como as telecomunicações, a medicina e até mesmo várias vertentes artísticas estão em constante transformação devido ao crescimento do leque de soluções e ao aparecimento de novas oportunidades no que respeita ao uso de novos equipamentos. É na aposta do uso de tecnologia na música que este projecto se foca. Nesta vertente artística têm surgido estilos musicais inovadores, novas técnicas associadas aos instrumentos e inevitavelmente diversos avanços no âmbito da instrumentação de som profissional. Com o intuito de inovar a abordagem musical têm surgido várias ideias que envolvem a projecção de imagens, jogos de luzes e movimentação do som, tudo de forma a tornar os espectáculos mais atraentes, elaborados e originais. Técnicas como a deslocação do som são também usadas noutras áreas como a do mundo cinematográfico (*surround* nos cinemas ou nos *home theaters*).

Com os avanços tecnológicos aparecem diversos sistemas que permitem a execução de várias ideias assim como o surgimento de novas. Na música electroacústica têm sido divulgados sistemas multicanal de grande escala que permitem a difusão do som conforme a vontade do artista tornando assim, tanto os espectáculos mais dinâmicos e enriquecidos como as próprias composições, que são muitas vezes feitas já a considerar a utilização destes sistemas. É debruçado neste assunto que surge este projecto. Existem já alguns sistemas multicanal de difusão de som que permitem a criação de diferentes cenários sonoros usando a movimentação do áudio através de vários altifalantes. O termo chave é *espacialização do som.*

É possível criar um sistema multicanal de difusão sonora com um tempo de desenvolvimento reduzido e de baixo custo cuja forma de implementação seja inovadora e vantajosa em relação às já existentes?

1.2 – Motivação

Este projecto surgiu de um desafio proposto pelo Prof. João Pedro Oliveira do Departamento de Comunicação e Arte (DECA) da Universidade de Aveiro. Sendo conceituado e detentor de vários prémios na área da música electroacústica, sugeriu a construção de uma matriz digital da mistura de áudio de forma a criar um sistema multicanal que permita a movimentação espacial do som com a possibilidade de ser posteriormente instalado num auditório do DECA.

Existindo poucos mecanismos que tenham um funcionamento semelhante, este projecto pretende inovar quanto ao método de implementação deste tipo de sistemas e trazer um produto que sirva a Universidade de Aveiro assim como um bom contributo à música electroacústica.

1.3 – Objectivos

Pretende-se neste projecto construir um sistema áudio multicanal de oito entradas analógicas e trinta e duas saídas, igualmente analógicas. Este sistema deverá realizar o balanceamento de todas as entradas pelas diversas saídas. A mistura deverá ser feita em tempo-real e a informação que a controla tem de ser enviada através de um PC. Dada a complexidade do sistema devido ao número de canais envolvidos e à necessidade de o processamento ser feito em tempo-real, a implementação deste projecto será feita com recurso a uma *Field Programmable Gate Array* (FPGA) [18] de forma a que a mistura seja inteiramente feita por hardware.

O sistema terá de ter a capacidade de repartir com o peso desejado, que poderá ser nulo, cada uma das entradas por cada uma das saídas, independentemente. Dado que os sinais em causa são sinais áudio, a pureza do sinal fornecido à entrada necessita de ser preservada, assim como o desfasamento entre os sinais. A Figura 1 exemplifica a constituição do sistema a implementar.

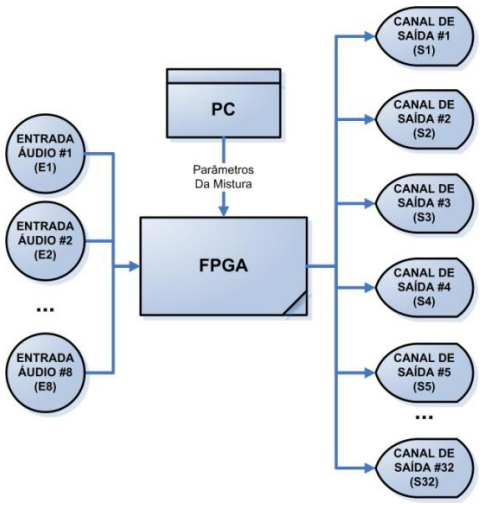


Figura 1 - Esquemático Simplificado do Sistema

1.4 – Organização da Dissertação

Esta dissertação refere-se à implementação de uma matriz digital de mistura de áudio e está dividida em seis capítulos servindo o primeiro de introdução e de contextualização do trabalho.

O Capítulo 2 refere a existência de sistemas multicanal de difusão sonora cujo conceito é semelhante ao que aqui se pretende apresentar. O conceito de espacialização de som é explicado e alguns sistemas são apresentados sendo o seu funcionamento analisado assim como as suas vantagens e desvantagens.

O Capítulo 3 descreve a forma de implementação do projecto aqui desenvolvido (MIAUDIO) abordando as suas principais características assim como os algoritmos que descrevem o seu funcionamento. O grande foco desta secção está centrado na descrição da solução em *Field Programmable Gate Array* (FPGA).

O Capítulo 4 procura descrever de forma mais detalhada todo o hardware envolvido neste projecto justificando as escolhas feitas, analisando o seu funcionamento e descrevendo as suas interligações.

De seguida, no Capítulo 5, são apresentados testes e resultados finais do sistema. Por último, no Capítulo 6, é feita uma conclusão e são apresentadas possíveis hipóteses de trabalhos futuros incidentes neste projecto.

Capítulo 2 – SISTEMAS MULTICANAL DE DIFUSÃO SONORA

Uma vez que existem sistemas semelhantes ao que se pretende desenvolver, é conveniente dar lugar a uma breve descrição de alguns deles. Serão incluídos alguns detalhes a nível de software assim como de hardware para que o funcionamento geral de sistemas deste tipo seja compreendido.

O Reino Unido tem uma forte presença nesta área sendo possível conhecer, em Inglaterra, o BEAST [22] [21] (Birmingham) e na Irlanda do Norte, o SARC [26] (Belfast). A nível de ferramentas de software existem duas bases importantes que suportam alguns dos sistemas mencionados. *Pro Tools* [25], usado no SARC, e *SuperCollider* [27][28], usado no BEAST, fazem parte de muitas das decisões tomadas pelos criadores destes sistemas. Será também descrita brevemente uma ferramenta denominada por MAX/MSP [24] uma vez que é uma aplicação muito usada no que respeita à composição de música electroacústica. Serão mais à frente apresentados alguns detalhes dos sistemas da universidade de Birmingham assim como a de Belfast.

Em primeiro lugar será explicado em que consiste de facto um sistema multicanal de difusão sonora.

2.1 – Sistema de Difusão Sonora – O que é?

Nos dias de hoje, em qualquer sala de espectáculos conceituada é possível encontrar uma distribuição criteriosa de altifalantes ao longo do espaço para que, durante os espectáculos, o espectador se sinta envolvido pela música e pelo som. É então necessário equipamento que permita a distribuição do som por vários altifalantes e que estes sejam posteriormente distribuídos pela sala. Até mesmo nos cinemas actuais, é perceptível que, usando algumas fontes sonoras, a movimentação do som é aproveitada para que o que se vê coincida com o que se ouve de forma a tornar o resultado mais realista. Por exemplo, num filme onde se tem um carro a passar de um lado do ecrã para o outro, o som deste elemento deve ser feito para que se ouça a chegar do mesmo lado com alguma intensidade, passe pelo meio (intensidade sonora igualmente distribuída pelos dois ouvidos) e que no fim som se desvaneça pelo lado em que o carro desaparece.

Em espectáculos musicais, principalmente na música electroacústica, estas técnicas de espacialização do som são muito aplicadas. Habitualmente usam-se diversas fontes sonoras que, ao serem colocadas estrategicamente, permitem criar vários cenários sonoros que representam uma imagem do som que se desloca ao longo da sala de espectáculos. Segue-se um exemplo simples com a utilização de dois canais de entrada a serem distribuídos por cinco saídas.

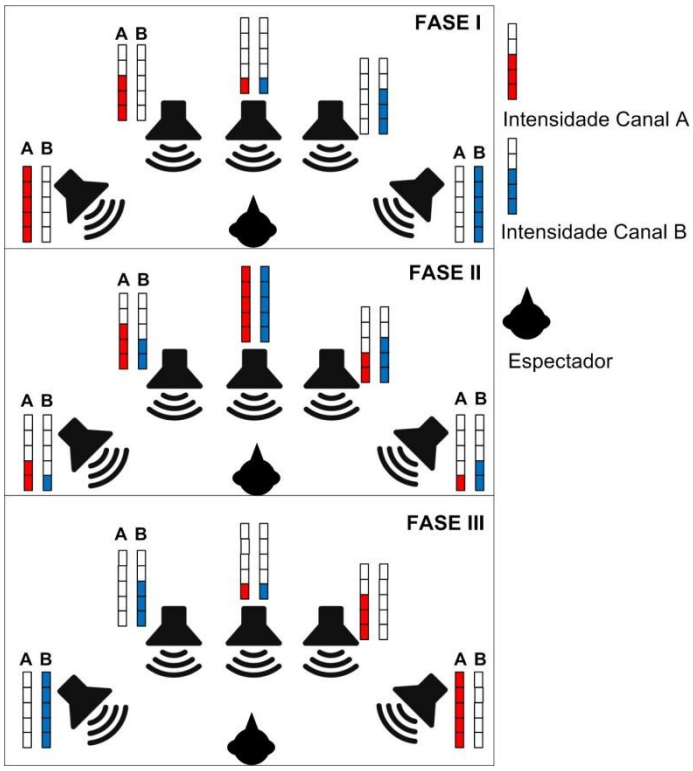


Figura 2 - Exemplo de Espacialização do Som

Como é intuitivo ao se observar a Figura 2 onde as barras identificam a intensidade de cada um dos canais (um a vermelho (A) e outro a azul (B)), se a passagem entre as três fases (I, II e III) for feita gradualmente, o ouvinte, estando posicionado frontalmente para as colunas, vai ter a sensação de que o som do primeiro canal, que se encontra inicialmente do seu lado esquerdo, se desloca para o lado direito enquanto que o segundo canal descreve a trajectória em sentido contrário. É esta a noção de espacialização do som e é este o método que os sistemas multicanal de difusão sonora que serão apresentados pretendem recriar. Nos sistemas mais complexos são utilizados mais canais, tanto de entrada como de saída.

A distribuição dos altifalantes desempenha um papel importante uma vez que é isso que vai definir a direcção de onde vem o som. A disposição mais comum segue o conceito de Jonty Harrison - *"Main Eight"* [20]. Como se pode observar pela Figura 3, são colocadas colunas em quatro zonas principais (*Main, Wide, Rear* e *Distant)*. As secções *Main* e *Wide* fornecem a imagem sonora principal e frontal. *Main* comporta-se como os altifalantes num estúdio normal e *Wide* é usado para alargar essa imagem, logo são colocados lateralmente. De seguida a secção

Rear, posicionada atrás da plateia, possibilita a imersão e a movimentação de 360º do som. Por último a secção *Distant* cria a percepção de distância em relação ao que é principal.

Resta apenas realçar que a movimentação do som é pensada já na fase de composição da música. Não é algo feito aleatoriamente durante a sua reprodução. Consoante a ideia que a obra quer transmitir, a deslocação é feita de forma adequada.

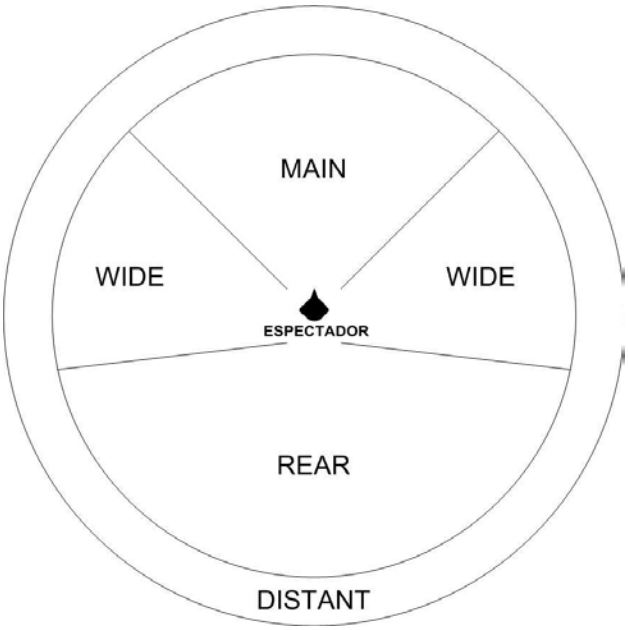


Figura 3 - Conceito de Jonty Harrison – "Main Eight"

2.2 – Sistemas Semelhantes Actuais

Serão agora apresentadas duas das várias estruturas existentes cujo conceito é semelhante ao que se quer criar neste projecto. Seguem-se algumas técnicas que são habitualmente utilizadas para permitir a criação de diferentes cenários sonoros:

- Uma gestão a nível de software;
- Interfaces com o utilizador via hardware;
- Estruturas móveis a sustentar as fontes sonoras;
- Posicionamento estratégico dos altifalantes.

2.2.1 – SARC – Sonic Arts Research Centre (Queens University, Belfast)

O SARC [26], desenvolvido e localizado em Belfast, é um sistema que permite uma experimentação exaustiva no campo da difusão do som uma vez que apresenta características inexistentes numa sala de espectáculos comum.

Este sistema é baseado numa ferramenta de software denominada como *Pro Tools HD3 Accel* (uma versão de *Pro Tools HD*, que será apresentada na secção 2.3 – Ferramentas de Controlo (Software)) e gere até 24 entradas analógicas que são estrategicamente distribuídas por vários altifalantes dispostos ao longo de 4 níveis. Estão instalados cerca de 112 altifalantes de forma a permitir uma grande flexibilidade na criação de diferentes cenários sonoros. As entradas são geridas com três mesas de mistura *Digidesign 192 I/O* [23] controlando assim 48 saídas analógicas. As mesas da *Digidesign* interagem com a aplicação *Pro Tools* para que a reprodução do som pelos diferentes altifalantes seja a desejada pelo utilizador.

Dada a constituição do sistema, este permite o desenvolvimento de estudos como o de posicionamento e *design* de alto-falantes e também na área de percepção e difusão do som.

Um detalhe importante é o facto de serem escolhidas colunas com uma projecção do som direccional. Isto é crucial uma vez que, tendo inúmeras fontes sonoras numa sala, isso torna o som mais inteligível e reduz as interferências entre as diferentes fontes. Na Figura 4 observa-se um esquema e uma breve descrição dos elementos base do auditório do SARC. A imagem que representa a distribuição dos altifalantes pela sala foi retirada de [26].

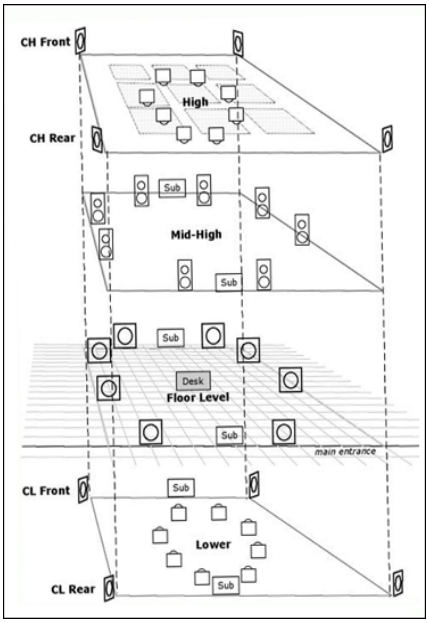
Esquema	Andar	Breve Descrição
	High	8 Colunas *Meyer*
	Mid-High	8 Colunas Meyer suspensas a 5m 1+1 Subwoofers suspensos
	Floor	Entrada 4 + 4 Colunas *Genelec* 1+1 Subwoofers
	Lower	8 Colunas *Meyer* 1+1 Subwoofers 8 Colunas *Meyer* suspensas 1 metro abaixo da entrada

Figura 4 - Esquemático da Sala do SARC

2.2.2 – BEAST – Birmingham ElectroAcoustic Sound Theater

O BEAST [21] [22] é outro sistema multicanal de grande escala criado para reprodução e inovação da música electroacústica. Foi desenvolvido na universidade de Birmingham e fundado em 1982. Desde então sofreu, como é de esperar, inúmeras alterações. Assim como o SARC, o BEAST proporciona a utilização de mais de 100 fontes sonoras podendo ser cada uma delas endereçada individualmente. Entre os vários alto-falantes existem alguns *arrays de tweeters* (altas frequências) suspensos sobre o público, *subwoofers* e outras colunas de diversos fabricantes. A distribuição das colunas segue o conceito *Main Eight* uma vez que Jonty Harrison, criador deste conceito, é o director deste projecto.

Inicialmente, o BEAST tinha um posto de controlo analógico em que uma mesa controlava a intensidade das colunas individualmente, por sectores ou grupos. As ligações eram directas logo, para alterações na estrutura, eram necessárias trocas de cabos e hardware. A inconveniência deste formato devido à tendência para problemas durante as actuações obrigou a uma reestruturação e introdução de uma matriz em que se balanceavam as entradas pelas diversas saídas. Este formato foi denominado como 3D. Mais tarde, com os requisitos a tornarem-se mais exigentes e com o avanço da tecnologia a possibilitar outras opções, o BEAST cedeu ao mundo do software. Inicialmente foi usado o MAX/MSP e agora o *SuperCollider*. O interface deste sistema é feito opcionalmente por uma mesa de fins dedicados ou por um computador. De momento o utilizador recorre a uma mesa de *faders Musical Instrument Digital Interface* (MIDI) reconfiguráveis. As pesquisas associadas ao BEAST relacionam-se com a composição electroacústica, sistemas electroacústicos ao vivo e interactivos e, composição e apresentação em sistemas multicanal de grande escala.

A Figura 5 representa algumas imagens do auditório do BEAST onde se destacam as numerosas fontes de reprodução assim como a estrutura onde se assentam.

Figura 5 - Birmingham ElectroAcoustic Sound Theater

2.3 – Ferramentas de Controlo (Software)

Dada a complexidade que um sistema de mistura e modulação de áudio pode atingir, o software é uma parte essencial do processamento de som quando é adoptada uma solução por software. A grande preocupação deste tipo de sistemas é o facto de o processamento ter de ser feito, na maior parte dos casos, em tempo-real. Assim sendo, as ferramentas apresentadas têm em conta essas considerações e tiram proveito do poder de processamento e da organização de sistemas operativos como o do MAC OS. As três ferramentas de softwares apresentadas nesta secção são por vezes usadas em sistemas como o BEAST e SARC.

2.3.1 – SuperCollider

O *SuperCollider* [27] [28] foi a aplicação escolhida pelos criadores do BEAST. Esta ferramenta é gratuita, corre apenas em MAC OS e usa uma linguagem de programação orientada por objectos própria que se assemelha a *Smalltalk* e C++ mas com uma sintaxe diferente. Esta linguagem inclui outras características que facilitam e alargam as funcionalidades do tratamento de som.

O *SuperCollider* tem como objectivo gerar e processar áudio. Este processamento pode ser feito em tempo-real se assim for desejado. A estrutura da linguagem de programação usada proporciona também a criação de algoritmos de composição. Os sinais áudio são obtidos por *streaming* através das portas usadas pelo gestor de som do MAC. Como interface com o utilizador para gerar o processamento em tempo-real, é possível configurar o software de forma que possa ser controlado por vários dispositivos como componentes com protocolo MIDI, rato, *graphics tablet* e *Open Sound Control*.

2.3.2 – Pro Tools

O *Pro Tools* [25] é uma plataforma usada em MAC OS ou em Windows que permite a gravação, importação e edição de música em vários formatos digitais. Esta aplicação pode vir acompanhada de hardware que permita satisfazer as necessidades do consumidor. Criado pela *Digidesign* (uma divisão da *Avid Technology*), este conjunto pode vir actualmente em três tipos de sistemas: HD, LE e M-Powered.

High-End (HD) é uma opção que integra software e hardware. Neste caso tem-se todo o processamento áudio executado em *Digital Signal Processors* (DSP) integrados na placa de som. A opção HD inclui também conversores analógico-digitais e *Peripheral Component*

Interconnect (PCI) interno. A comunicação entre os dispositivos que fornecem o sinal áudio é gerida por multiplexagem no tempo. Ao escolher o *Pro Tools HD* tem-se geralmente 16 entradas/saídas e informação analógica e/ou digital. A frequência de amostragem difere em dois subtipos diferentes, podendo ser de 96kHz ou 192kHz, no máximo.

A grande diferença de HD para LE reside no facto de todo o processamento de um sistema LE ser feito no CPU local, o que representa uma grande desvantagem. A relação entrada/saída é gerida por um dispositivo USB da *Digidesign* ou por um interface *Firewire*.

Finalmente, o *Pro Tools M-Powered* obtém o seu nome após a compra da *M-Audio* por parte da *Avid Technology*. Após esta junção, a *Digidesign* e a *M-Audio* começaram a desenvolver o *Pro Tools M-Powered* que hoje engloba quase todas as funcionalidades do *Pro Tools LE*. É esta a versão que tem sido aperfeiçoada e que agora permite a personalização de sistemas de multicanais compactos em Windows XP e MAC OS.

O interface com utilizador do *Pro Tools* tenta encontrar um meio-termo entre as mesas de mistura analógicas e as mais recentes estações digitais de áudio tendo portanto desde *faders* analógicos a painéis de controlo tácteis. Muito do hardware disponibilizado pela *Digidesign* permite uma interacção com esta ferramenta.

2.2.3 – MAX/MSP

O MAX/MSP [24] é igualmente uma aplicação muito usada em ambiente MAC. Desenvolvido pela *Cycling'74* esta ferramenta, assim como o *SuperCollider*, utiliza programação por objectos. São então usados vários módulos de programação, sendo muitos deles fornecidos no pacote inicial e outros criados e partilhados entre utilizadores, que permitem a sua interligação e formação de blocos mais complexos.

Esta aplicação possui duas componentes. MAX é a parte responsável pelo ambiente de programação gráfico que providencia um interface ao utilizador assim como comunicação e suporte MIDI. A segunda parte é o MSP que é a componente encarregue pelo processamento digital do sinal e pela síntese do áudio em tempo-real. Esta ferramenta assenta o seu funcionamento nesta estrutura desde 1997.

Esta aplicação tem um papel importante no projecto em desenvolvimento uma vez que será uma possível escolha para a interface da matriz de áudio.

2.4 – Comparação e Conclusões

O sistema BEAST e o SARC, apesar de terem implementações distintas, são bastante semelhantes na forma como geram a informação áudio. Ambos os sistemas têm uma aplicação que, ao fazer a mistura dos sinais de entrada, permite gerar o sinal de saída. O hardware usado nestes sistemas serve para criar um interface com o utilizador para que este tenha controlo sobre a mistura a ser executada pela ferramenta de software. Assim sendo, a mistura é realizada por software em que este usa informação interna ou externa para definir os parâmetros da mistura.

Por sua vez, o sistema em desenvolvimento neste projecto (MIAUDIO) tem de certa forma os papéis invertidos. Os sinais áudio adquiridos são misturados por um sistema digital reconfigurável, a FPGA, ou seja, a mistura é feita em hardware. O software nesta implementação tem a responsabilidade de definir os parâmetros da mistura. Dado que esta função requer muito pouco tempo de processamento, o computador que controla a gestão do som fica livre para desempenhar muitas outras funções como criar uma interacção do áudio com vídeo, aplicar efeitos ou outros processamentos nos canais que são enviados para o sistema, entre outras. Este é um aspecto diferenciador em relação aos sistemas que adoptam uma solução por software uma vez que essa implementação fica fortemente dependente do sistema operativo (SO). É necessário um SO eficiente e fiável mas acima de tudo, com um poder de processamento elevado. Em MIAUDIO, a forma como a mistura é feita é livre podendo recorrer-se a programas para enviar os parâmetros para o hardware ou a interfaces como as mesas usadas no BEAST e no SARC.

Se for necessário introduzir novas funcionalidades, na solução escolhida no sistema em desenvolvimento, o hardware pode permanecer inalterado e as funcionalidades que necessitam de ser acrescentadas podem ser feitas num nível mais alto, ou seja, no sistema operativo que gere o balanceamento dos canais áudio. É também possível acrescentar hardware extra que interaja com o sistema operativo. Mesmo sendo necessário alterar o algoritmo que corre na FPGA, esta pode ser reconfigurada facilmente e sem implicar alterações a nível de hardware.

No sistema MIAUDIO destaca-se o facto de o sistema desenvolvido ter um custo associado reduzido e um tempo de desenvolvimento relativamente curto. Em anexo (Anexo C – Custo do Sistema) é possível encontrar os detalhes dos custos do sistema.

Apresenta-se agora uma tabela que resume brevemente e compara algumas características dos dois sistemas acima descritos juntamente com MIAUDIO:

	BEAST	**SARC**	**MIAUDIO**
Software de Suporte	SuperCollider	Pro Tools	*A definir*
Tipo de Entrada/Saída	Analógico/Analógico	Analógico/Analógico	Analógico/Analógico
Número de Entradas/Saídas	*Desconhecido*	24/48	8/32
Fontes sonoras (reprodução)	+100	112	*A definir*
Interface com Utilizador	Faders MIDI Reconfiguráveis	Mesa de Controlo da Digidesign	*A definir*
Realização da Mistura	Software	Software	Hardware
Controlo da Mistura	Hardware	Hardware	Software

Tabela 1 - Comparação BEAST vs. SARC vs MIAUDIO

Os dois sistemas discutidos, BEAST e SARC, apresentam um potencial semelhante dado que têm sensivelmente a mesma dimensão e que o método de implementação é idêntico apesar de usarem tanto software como hardware diferente. Ambos sistemas são inovadores e trouxeram um grande contributo para os espectáculos de música electroacústica.

Capítulo 3 – MATRIZ DIGITAL DE MISTURA DE ÁUDIO

3.1 – Descrição do Sistema Implementado

Pretende-se que o sistema desenvolvido seja capaz de adquirir oito canais áudio analógicos e misturá-los em tempo-real de forma a balancear cada uma das entradas, independentemente, pelas trinta e duas saídas possíveis. Esta topologia pode ser interpretada como uma matriz em que cada parâmetro define a intensidade de um canal de entrada numa determinada saída. A informação que define o peso de cada uma das entradas em cada uma das saídas é enviada do computador para uma *Field Programmable Gate Array* (FPGA) onde é feito o processamento do sistema. Esta matriz de mistura é definida por 256 parâmetros ou coeficientes, isto é, se cada canal de entrada pode ter um peso diferente em cada uma das saídas e, se o sistema é constituído por 8 entradas e 32 saídas, resultam 8 × 32 coeficientes.

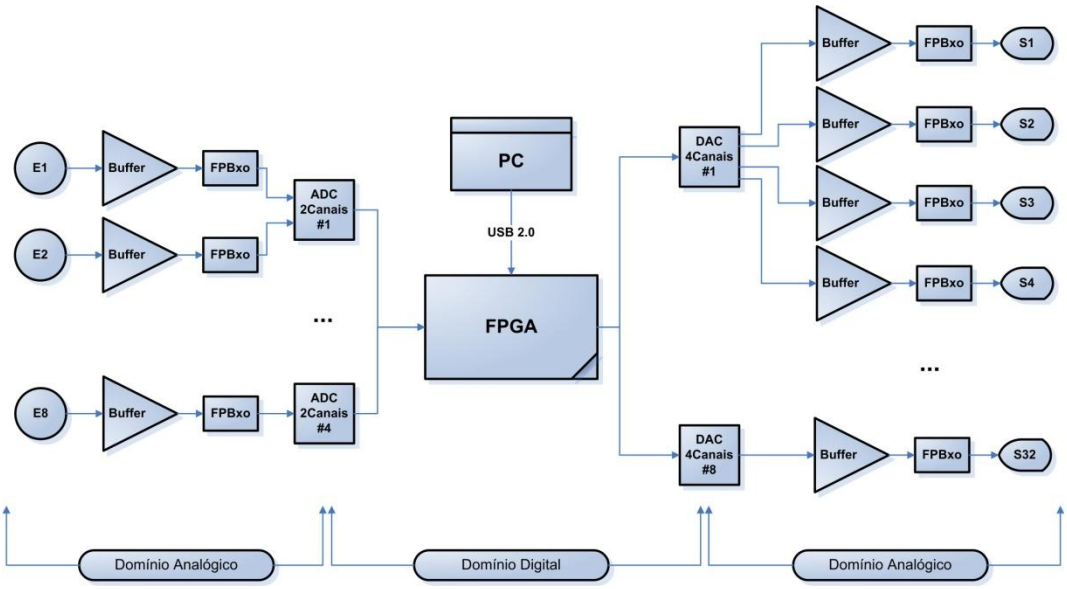


Figura 6 - Esquema do Sistema Implementado

Dado que os sinais à entrada do sistema são analógicos, é necessário hardware que torne possível o processamento por parte da FPGA. Como se pode observar pela Figura 6, o sistema actua com o exterior analogicamente mas no entanto o seu processamento interno é todo digital. Existe hardware responsável pelo acondicionamento analógico e são usados conversores analógico-digital (ADC) e digital-analógico (DAC) para criarem uma ponte entre os dois domínios, analógico e digital. O andar de entrada é constituído por um buffer por canal seguido

de um filtro passa-baixo. Os vários sinais são colocados na entrada (E), passam pelo hardware responsável pelo seu acondicionamento, são de seguida amostrados pelos ADCs e finalmente enviados para a FPGA.

Assim que a FPGA tiver recebido a primeira amostra de cada um dos canais, esta desencadeia o seu processamento interno. A Figura 7 representa, de uma forma simplificada, os blocos que constituem a lógica da FPGA. O Bloco de Entrada é concebido para interagir com os conversores analógico-digital. Os canais recebidos são enviados para o Bloco Aritmético onde é desenvolvida a matriz que descreve a mistura áudio. Esta mistura necessita também da informação enviada pelo PC. O Bloco de Comunicação USB gere esta transferência de dados e armazena esta informação com auxílio do Bloco de Controlo de Memória. Desenvolvida a matriz, os sinais resultantes são convertidos para um formato adequado aos conversores digital-analógico e são de seguida enviados para o Bloco de Saída. Este último cria um interface com os conversores digital-analógico.

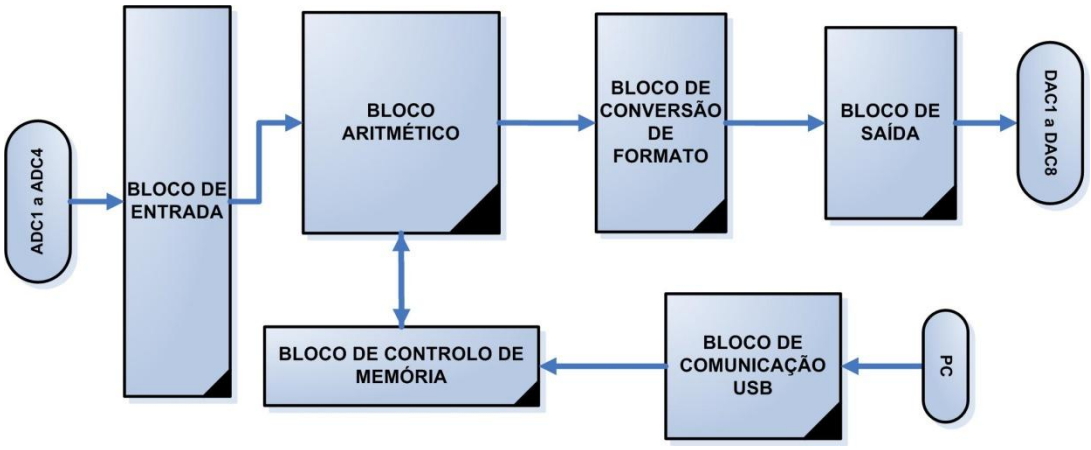


Figura 7 - Blocos Internos da FPGA

O Bloco Aritmético é o bloco mais complexo dado que é aqui que é definida a matriz de mistura do sistema. O algoritmo de mistura é desenvolvido de modo a se realizarem todos os produtos entre os canais recebidos (E) e os respectivos coeficientes de forma a serem calculadas todas as saídas (S). A Figura 8 descreve precisamente esse comportamento. Aqui é possível observar a forma como se relacionam as saídas do sistema com as entradas.

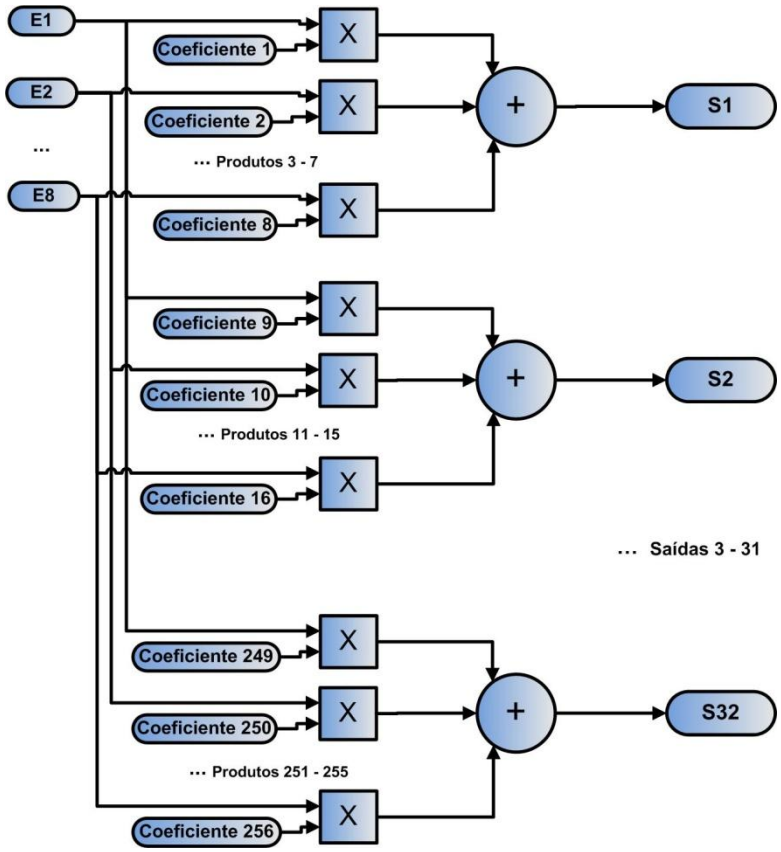


Figura 8 - Esquema do Bloco Aritmético

Como se pode observar pela Figura 8, cada um dos sinais de entrada vai ter de ser usado em 32 produtos diferentes uma vez que existe, para cada saída, um coeficiente diferente que define o peso dessa entrada em cada um dos 32 canais. Para cada saída são somados os oito produtos que correspondem à multiplicação de cada um dos oito canais pelos respectivos coeficientes. Realizam-se então 256 multiplicações e 32 somas de oito parcelas. Para tornar as multiplicações mais eficientes, são usados multiplicadores dedicados presentes na FPGA.

Assim que os sinais de saída são calculados, a FPGA inicia a comunicação com os DACs, respeitando o protocolo implementado para que seja dado lugar à conversão de digital para analógico. Os sinais enviados pela FPGA, após serem convertidos para analógico, são filtrados passa-baixo e colocados na saída com recurso a buffers.

3.2 – Constituição do Sistema

Compreendido o funcionamento genérico do sistema é conveniente perceber como é que todos os componentes que foram mencionados se interligam. A Figura 9 representa um bloco de entrada e outro de saída.

O sinal áudio é ligado na entrada por intermédio de uma ficha XLR [29] que o entrega em formato diferencial. Este sinal passa de seguida por um buffer de entrada, o INA137 [4], que o converte de diferencial para unipolar. O filtro passa-baixo (FPBxo) que se apresenta de seguida, criado usando amplificadores operacionais MCPs [13], serve para retirar informação contida em frequências fora da banda de interesse. Na entrada são colocados os amplificadores MCP602s e na saída MCP604s. O primeiro integrado tem dois amplificadores operacionais embutidos enquanto que o segundo tem quatro. Tendo os sinais acondicionados, estes são amostrados pelos ADCs (PCM1802 [3]) e enviados para a FPGA que interpreta a informação de acordo com o protocolo implementado entre os dois elementos. O oscilador representado na figura é comum a todos os conversores analógico-digital para que a amostragem seja simultânea. O ADC, usando o sinal do oscilador, gera internamente o relógio que controla a linha de envio série sendo portanto, juntamente com outros sinais de controlo que serão apresentados na descrição do PCM1802, ligado à FPGA. Cada ADC é responsável por amostrar dois canais de entrada.

No bloco de saída ocorre sensivelmente o processo inverso. A FPGA, após ter processado internamente os sinais de entrada juntamente com os coeficientes da matriz e gerado as respectivas saídas, envia para cada um dos DACs (DAC8534 [5]) quatro canais. Neste caso, o envio série é controlado por um relógio criado pela FPGA. Este sinal é enviado, assim como os de controlo necessários para a configuração dos conversores digital-analógico, para todos os DACs do sistema. O DAC converte os quatro canais que lhe foram entregues e coloca-os na saída. Os sinais resultantes são primeiro filtrados e de seguida convertidos de unipolar para diferencial pelos buffers de saída DRV134 [2] que assim tornam a saída do sistema balanceada [19] e adequada à reprodução e integração em sistemas áudio.

Dados os relógios que controlam ambos os envios, os conversores analógico-digital funcionam em modo *Master* e os DACs em modo *Slave.* A FPGA fica assim a aguardar a recepção de amostras, processa-as assim que as recebe, e fica em espera até que cheguem as seguintes. O processamento interno é obrigatoriamente feito no espaço de tempo entre amostras.

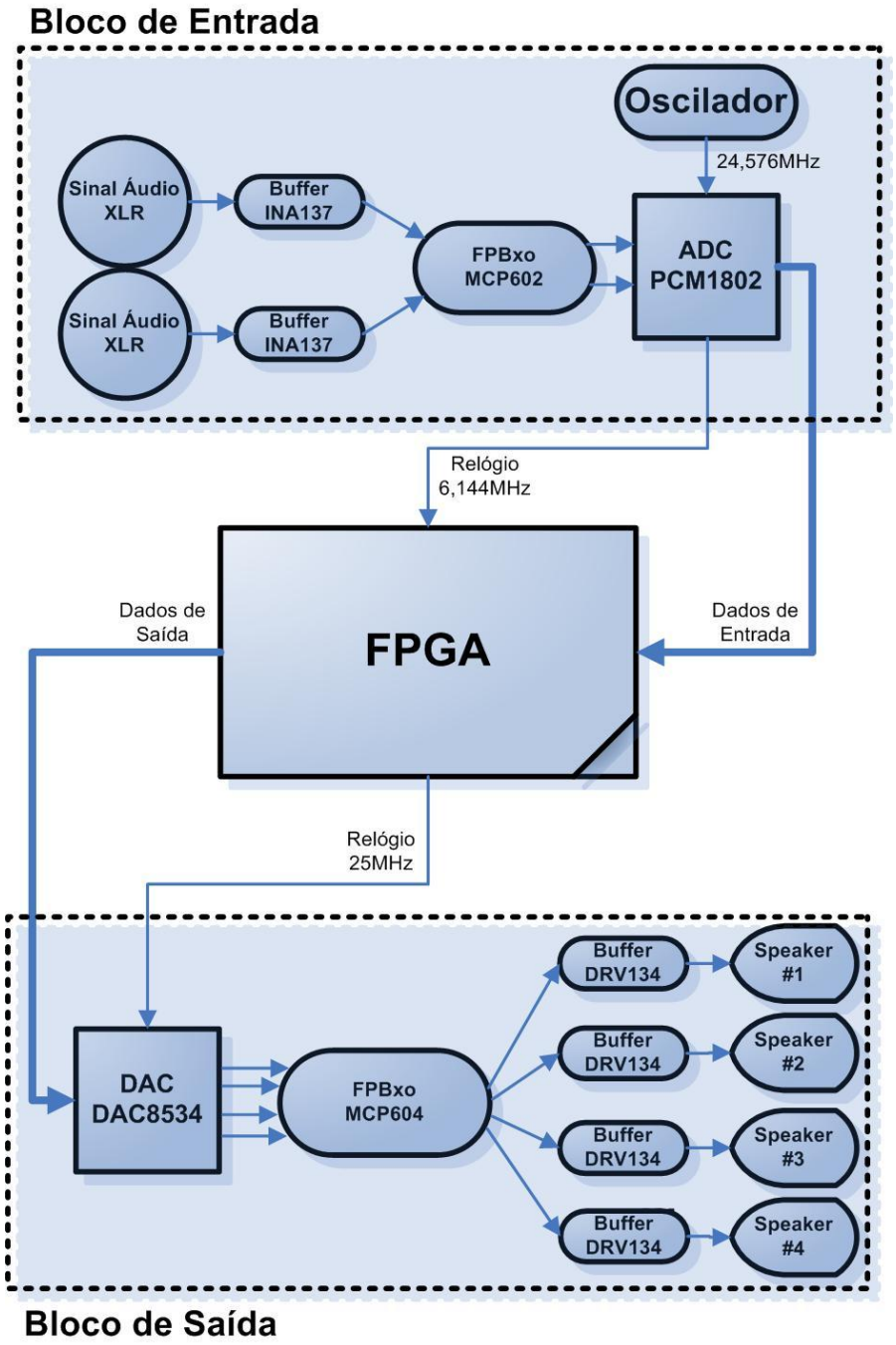


Figura 9 - Interligação dos Componentes na Entrada e Saída

Os blocos de entrada e de saída foram replicados até se obter o número de entradas e saídas desejadas. São necessários quatro conjuntos de hardware que envolvem os ADCs e oito para o lado dos DACs. Obtêm-se assim as oito entradas e trinta e duas saídas. Contabilizam-se então quatro MCP602, oito INA137, oito MCP604 e trinta e dois DRV143.

3.3 – Lógica Interna da FPGA

A FPGA presente na placa de desenvolvimento da Digilent® tem de ser configurada de forma a que:

- Receba informação dos conversores analógico-digital (ADC) colocados na entrada do sistema;
- Comunique com o computador através da ligação USB existente de forma a receber os parâmetros da matriz;
- Processe o algoritmo aritmético que descreve o comportamento da matriz de mistura de áudio;
- Configure e envie informação para os conversores digital-analógico (DAC) que se encontram na saída.

Em primeiro lugar observa-se um diagrama, Figura 10, que representa os vários blocos que constituem a lógica interna da FPGA assim como o seu interface com os ADCs, DACs e PC. Vista a interacção entre cada um deles, será dada uma descrição mais detalhada sobre o funcionamento de cada um dos blocos individualmente.

Durante a implementação do algoritmo foram tomadas várias precauções. Tentou-se minimizar a lógica sintetizada de forma a se utilizar o mínimo de recursos possíveis para que o algoritmo seja eficiente e consequentemente reduzir o consumo total. Dada a complexidade e a quantidade de lógica envolvida foram tomados vários cuidados no que diz respeito aos sinais que servem de relógio para o hardware sintetizado. Dada a existência de sinais provenientes dos ADCs que servirão de relógio para parte da lógica implementada, e que a restante é controlada com recurso ao relógio interno da FPGA foram tomadas precauções adicionais como se poderá ver na descrição do bloco de entrada.

Como se observa pela Figura 10, a lógica da FPGA é composta por vários blocos internos que desempenham, cada um deles, uma função específica. O Bloco de Entrada tem como objectivo a interpretação dos sinais provenientes dos vários conversores analógico-digital para que as amostras sejam recebidas correctamente. Uma amostra de cada canal é armazenada até que todos os conversores tenham feito o envio dos dois canais de entrada a que estão ligados. Assim que isto suceder, o processamento destas amostras é desencadeado.

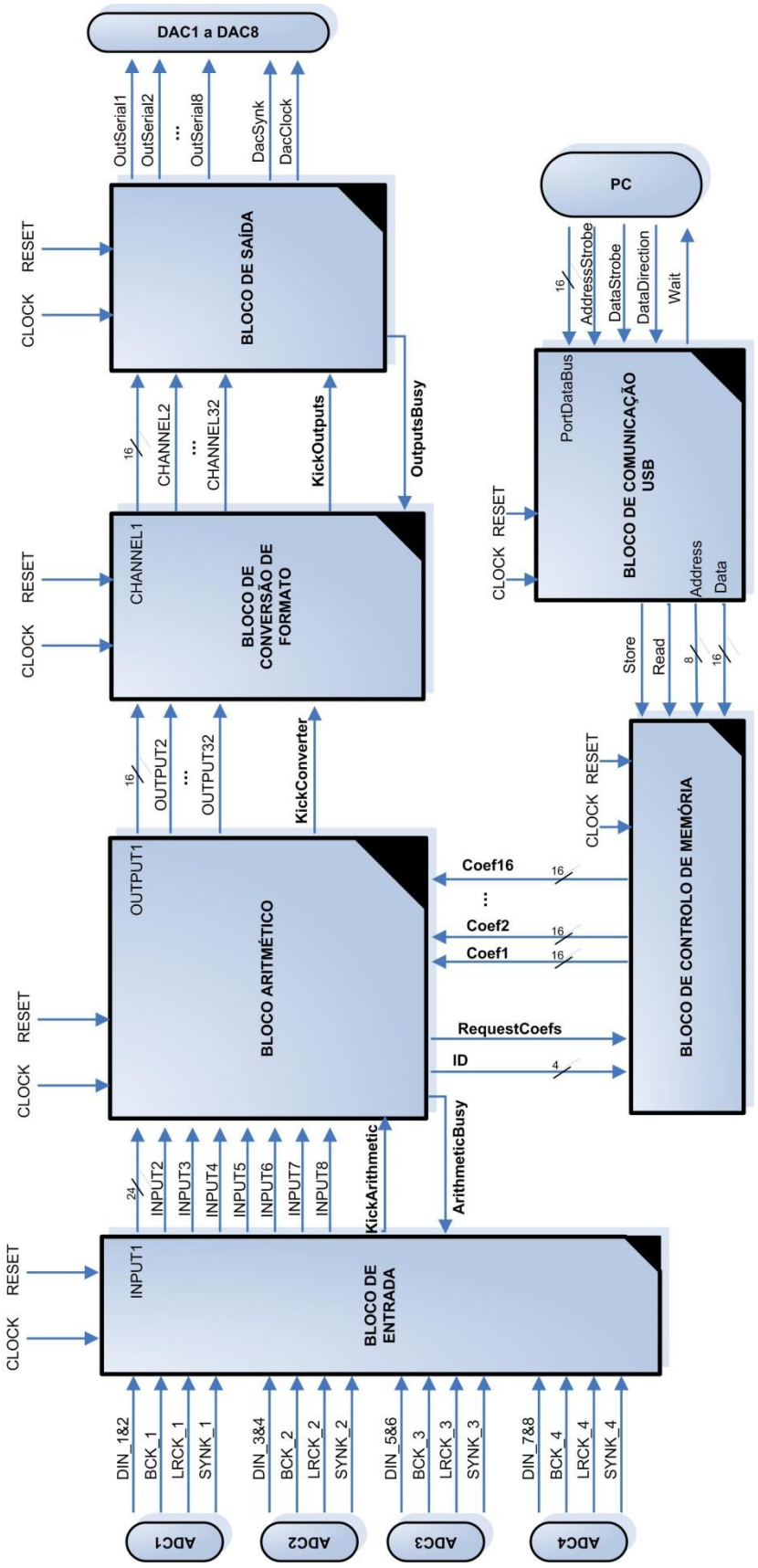


Figura 10 - Lógica Interna da FPGA

O bloco seguinte, Bloco Aritmético, é o bloco mais complexo dado que contém o algoritmo central que faz a mistura dos sinais representado na Figura 8. Este bloco interage com o bloco de Entrada, de onde recebe os sinais de entrada, e com o bloco responsável pelo controlo da memória, onde estão armazenados os coeficientes da matriz que foram enviados pelo computador.

O bloco denominado como Controlo de Memória tem a responsabilidade de gerir os bancos de memória embutidos na FPGA (Block RAMs) onde são armazenados os parâmetros da matriz. Este bloco tem como função controlar o seu acesso e guardar informação quando a comunicação USB assim o solicita. A leitura dos bancos de memória e o armazenamento dessa informação em registos é feito quando existe um pedido por parte do Bloco Aritmético. O número de bancos de memória necessários assim como informação relevante sobre estes componentes vai ser apresentada mais à frente.

A lógica que controla a comunicação USB (Bloco de Comunicação USB) foi disponibilizada pela Digilent® e adaptada consoante as necessidades deste projecto. A sua função, após a adaptação, é receber a informação via USB e pedir ao bloco que gere os bancos de memória dedicados que a armazene de forma adequada.

O penúltimo bloco é responsável pela conversão de formato dos sinais que foram gerados no Bloco Aritmético. Todos os sinais da matriz foram até este ponto tratados em complemento para dois uma vez que foi neste formato que os ADCs enviaram os dados e também devido ao facto de os multiplicadores dedicados usados no bloco Aritmético necessitarem que a informação esteja em complemento para dois. Por sua vez, como os DACs requerem que essa informação seja enviada em binário natural, logo, este bloco faz a ponte entre os dois formatos. Feita essa conversão, os sinais finais são enviados para o bloco de Saída. Este bloco interage com os conversores digital-analógico gerando os sinais de controlo adequados para a sua configuração e para a conversão dos dados enviados.

Os sinais representados na figura como *Kick* são usados para sinalizar, após cada bloco ter completado a sua função, o bloco seguinte para que o processamento continue. Existem também os sinais *Busy* que identificam o estado actual do bloco, podendo este estar livre ou ocupado. Este sinal de controlo existe apenas nos blocos mais complexos uma vez que o seu tempo de processamento é superior. O sinal de relógio é comum a toda a lógica implementada, à excepção da lógica que interage com os ADCs. Por sua vez, o sinal de reset é comum a toda a lógica criada. Neste diagrama é possível observar que alguns sinais são barramentos cujo número de bits está representado sobre a seta. Os sinais idênticos têm o mesmo número de bits, por exemplo, *Input1* e *Input2*. As linhas sem indicação do número de bits, são de apenas um bit.

3.3.1 – Recepção das Entradas: Interface com os ADCs

O Bloco de Entrada, como foi anteriormente referido, está encarregue de comunicar com os conversores analógico-digital. O diagrama temporal apresentado na Figura 11 representa os sinais de funcionamento do ADC. Este é configurado como *Master*, ou seja, controla a conversão e o envio da informação. A lógica gerada na FPGA tem então de interpretar os sinais representados na figura de forma a receber correctamente as amostras do sinal. O sinal *FSYNK* observado no diagrama, quando se encontra no valor lógico '1', identifica o intervalo de tempo em que a informação está a ser enviada. Essa informação pode ser referente a um dos dois canais, logo, o sinal *LRCK* distingue-os. Por sua vez, o sinal *BCK* serve de relógio de transmissão enquanto que a linha *DOUT* transporta os dados propriamente ditos.

A lógica sintetizada na FPGA permite implementar um *shift-register* que recebe a palavra que é enviada pelo conversor através da linha série *DOUT*, desde o bit mais significativo ao menos significativo. A máquina de estados finitos descrita no diagrama da Figura 12 fica em espera enquanto o registo de deslocamento está a ser actualizado e armazena os dados nele contidos quando o envio termina.

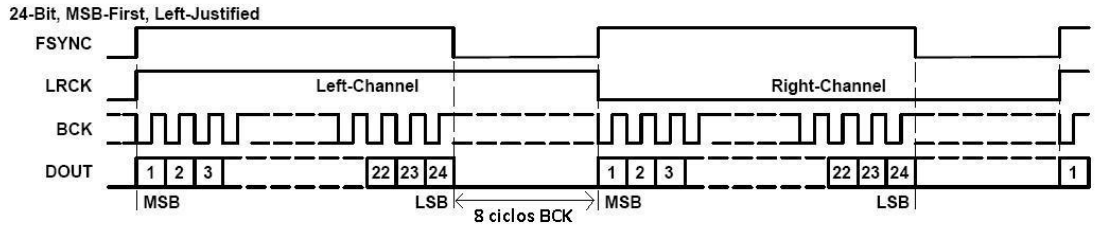


Figura 11 - Diagrama Temporal de Funcionamento do ADC

Para guardar as amostras recebidas foi instanciada uma fila First-in-First-Out (FIFO) disponibilizada pela fabricante da FPGA, Xilinx®, uma vez que esta permite a utilização de dois domínios de relógios, um para escrita e outro para leitura. Esta medida surge na sequência de existir um relógio de 6,144MHz gerado por cada conversor analógico-digital, *BCK*, que controla o envio dos sinais digitais e um outro relógio de 50MHz que controla a restante lógica da FPGA. O algoritmo de implementação da FIFO está optimizado e retira a preocupação de sincronização entre os dois domínios do projectista uma vez que este conflito é resolvido internamente. Assim sendo, instanciou-se uma FIFO para cada canal para que, quando uma amostra de um canal é recebida, esta seja armazenada na fila correspondente. A Figura 12 representa precisamente esse comportamento. Este bloco lógico é replicado consoante o número de ADCs usados sendo a sua relação de um para um. A Tabela 2 estabelece a correspondência entre o nome dos sinais, a sua descrição e o número de bits dos diagramas da Figura 12 e Figura

13. Esta tabela indica também se um sinal é recebido à entrada do bloco, se é usado internamente ou se é um sinal de saída.

A máquina de estados da Figura 12 é inicializada no estado *a0*, usa o sinal *FifoFull* para permitir um funcionamento correcto da FIFO e, se esta não se encontrar com todas as posições da fila ocupadas, passa para o estado *a1* no qual aguarda até que seja detectado, através de *FSYNK*, o fim de um envio por parte dos conversores. Quando isto acontece, a máquina identifica, avaliando *LRCK*, qual o canal enviado e vai buscar a palavra armazenada ao *shift-register* correspondente e introduz essa informação na fila apropriada. No estado *a4* ou *a5* é feita a desactivação da escrita na FIFO correspondente sendo que existem de seguida alguns ciclos de espera no estado *a7* até que outro envio seja iniciado. O relógio de escrita na FIFO é o sinal que controla o envio de dados dos ADCs (*BCK*) enquanto que o de leitura é o relógio interno da FPGA.

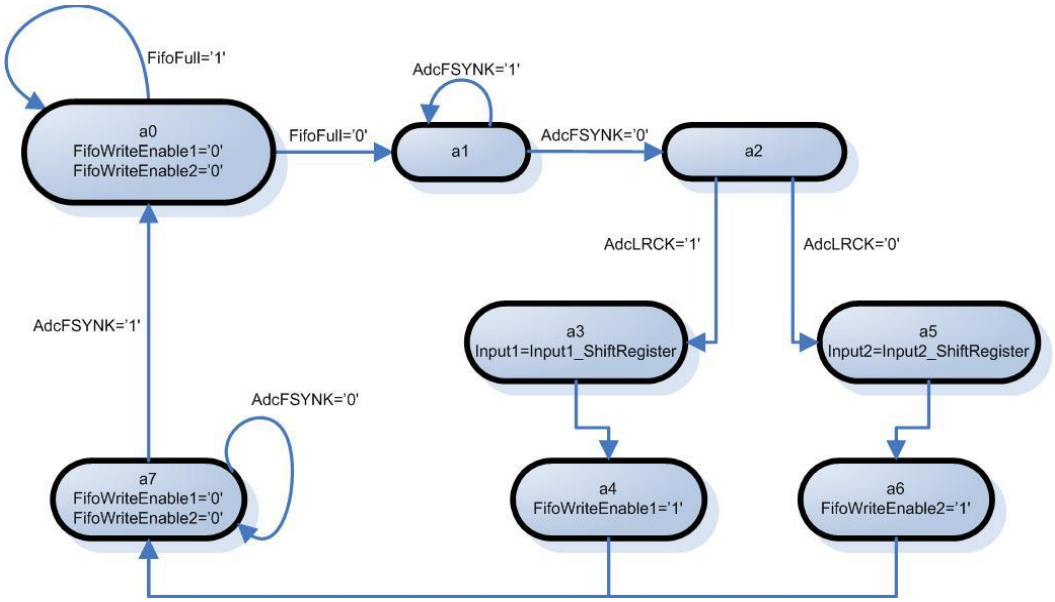


Figura 12 – Máquina de Estados: Entrada – Escrita na FIFO

A outra máquina de estados, Figura 13, implementada no Bloco de Entrada serve para avaliar se já foi enviada uma amostra de cada canal. Para tal, o sinal *FifoAlmostEmpty* de todas as FIFOs é avaliado e, quando se encontrarem todos activos, o respectivo conteúdo das filas é armazenado em registos que serão usados pelo bloco Aritmético. Neste ponto, terá já sido recebida uma amostra de cada um dos oito canais. No estado *a4* observa-se uma verificação do sinal *ArithmeticBusy* para que a sinalização de informação pronta não seja feita enquanto o bloco aritmético estiver em processamento. Este acontecimento levaria a uma perda das amostras em processamento. Esta medida é feita para salvaguardar o funcionamento do sistema mas no entanto nunca haverá um bloco ocupado enquanto o anterior lhe tenta dar a amostra

seguinte uma vez que o processamento das amostras é mais rápido que a taxa de amostragem dos ADCs.

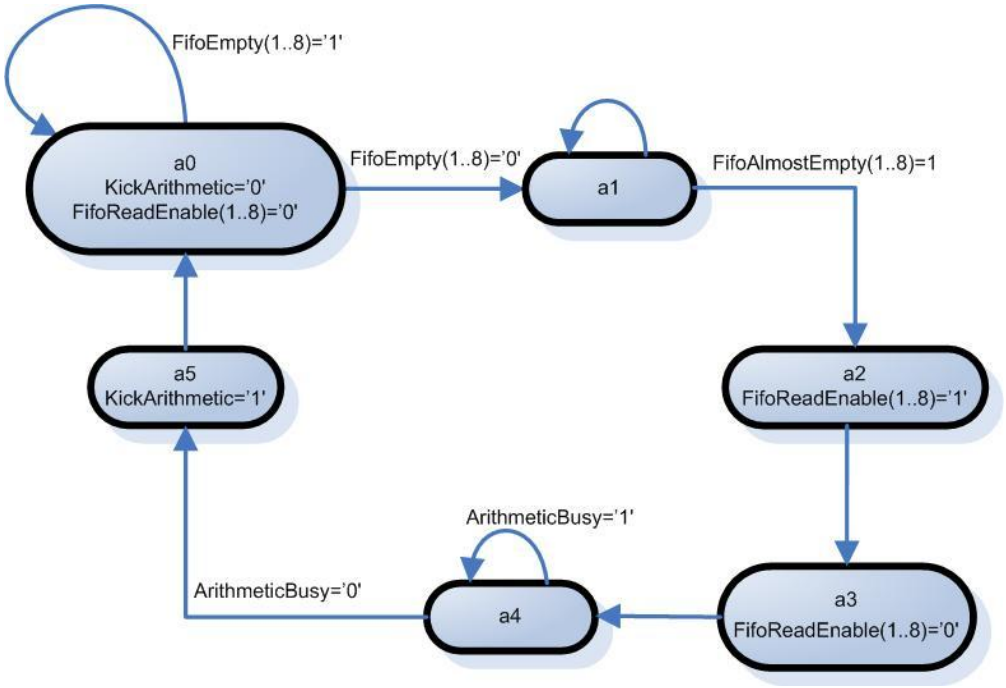


Figura 13 – Máquina de Estados: Entrada – Leitura da FIFO

SINAL	**DESCRIÇÃO**	**TIPO**	**Número de Bits**
FifoWriteEnable	Sinalização de Escrita na FIFO	Interno	1
FifoReadEnable	Sinalização de Leitura da FIFO	Interno	1
FifoFull	Sinaliza a FIFO cheia	Interno	1
FifoEmpty	Sinaliza a FIFO vazia	Interno	1
FifoAlmostEmpty	Sinaliza a FIFO com uma posição ocupada	Interno	1
AdcFSYNK	FSYNK proveniente dos ADCs	Entrada	1
AdcLRCK	LRCK proveniente dos ADCs	Entrada	1
Input	Registo de Armazenamento da Amostra	Interno	24
Input_ShiftRegister	Registo de Armazenamento do *ShifRegister*	Interno	24
ArtihmeticBusy	Sinalização do estado do bloco Aritmético	Entrada	1
KickArithmetic	Sinalização de informação pronta ao bloco Aritmético	Saída	1

Tabela 2 - Descrição dos Sinais: Entrada

3.3.2 – Bloco Aritmético: Desenvolvimento da Matriz

O bloco agora descrito é o bloco responsável pelo algoritmo da implementação da matriz de mistura. Tem então como função receber os oito sinais de entrada e, considerando os vários coeficientes que definem a matriz, gerar as trinta e duas saídas desejadas. Os sinais de entrada são entregues pelo Bloco de Entrada enquanto que os coeficientes da matriz têm de ser pedidos ao bloco que faz a gestão dos bancos de memória onde a informação recebida via USB é armazenada.

Como já foi referido, foram usados multiplicadores dedicados que permitem que esta operação seja eficientemente feita num ciclo de relógio. Estes multiplicadores têm como entrada dois registos de 18bits e geram uma saída de 36bits. Todos os operandos estão representados em complemento para dois sendo que os dois bits mais significativos da saída são ambos bits de sinal. Uma vez que a FPGA em uso tem apenas vinte multiplicadores dedicados optou-se por criar dezasseis ciclos onde se efectuam, em cada um, dezasseis produtos. Assim sendo, como em cada ciclo se usam dezasseis multiplicadores, ao fim dos dezasseis ciclos obtêm-se os 256 produtos que permitem obter as 32 saídas. Criou-se então uma máquina de estados cíclica que faz, em cada ciclo, a recolha dos operandos, actualização das entradas e respectiva recolha das saídas dos multiplicadores, acondicionamento dos resultados fazendo arredondamentos e detecção de overflow e armazenamento dos resultados nos registos adequados. Em cada ciclo existem registos auxiliares que são reutilizados ao longo do processo para minimizar a quantidade de lógica sintetizada. Cada ciclo completo gera duas saídas distintas uma vez que para cada saída são precisas apenas oito multiplicações dado que cada saída pode ter informação de qualquer uma das oito entradas.

A máquina de estados descrita no diagrama da Figura 14 fica no primeiro estado, *a0*, até que seja activo o sinal *KickArithmetic* por parte do bloco de Entrada. Quando isto acontece, as oito amostras de entrada são lidas para um registo no estado *a1* e, no seguinte, é feito o pedido, com auxílio à *flag RequestCoefs*, dos primeiros dezasseis coeficientes ao bloco que controla a memória embutida. O estado *a3* é um estado de espera uma vez que se quer garantir que os coeficientes já estão estabilizados nos registos respectivos. No ciclo de relógio seguinte é feita a actualização das entradas dos multiplicadores sendo depois armazenado o resultado para dezasseis registos de 36 bits. No estado *a6* é feita a expansão do sinal e o descarte dos bits menos significativos do produto para que a soma entre os oito registos seja feita e sujeita a arredondamento e detecção de overflow. Nos dois estados seguintes é feita a soma e o arredondamento, respectivamente. Os algoritmos de arredondamento e overflow serão descritos de seguida.

No estado *a8*, o sinal *IDcoefs* é analisado para que se saiba que duas saídas estão a ser calculadas. Assim sendo, a máquina salta para o estado correspondente onde é feita a detecção

de overflow e armazenamento dos resultados em dois dos trinta e dois registos de saída. Nos quinze primeiros ciclos a máquina regressa ao estado *a2* onde é feito o pedido dos próximos dezasseis coeficientes. Apenas no último ciclo, onde já foram calculadas todas as saídas, é dado o salto para o estado em que é feita a sinalização ao bloco seguinte, (bloco de Conversão de Formatos) através do sinal *KickConverter*, e de seguida o regresso ao estado *a0* onde a máquina permanecerá até que sejam recebidas mais amostras. O sinal *ArithmeticBusy* é usado para representar o estado em que o bloco Aritmético se encontra, podendo este estar livre ou ocupado.

A Tabela 3 apresenta uma descrição, assim como o número de bits dos sinais apresentados nesta máquina de estados finitos.

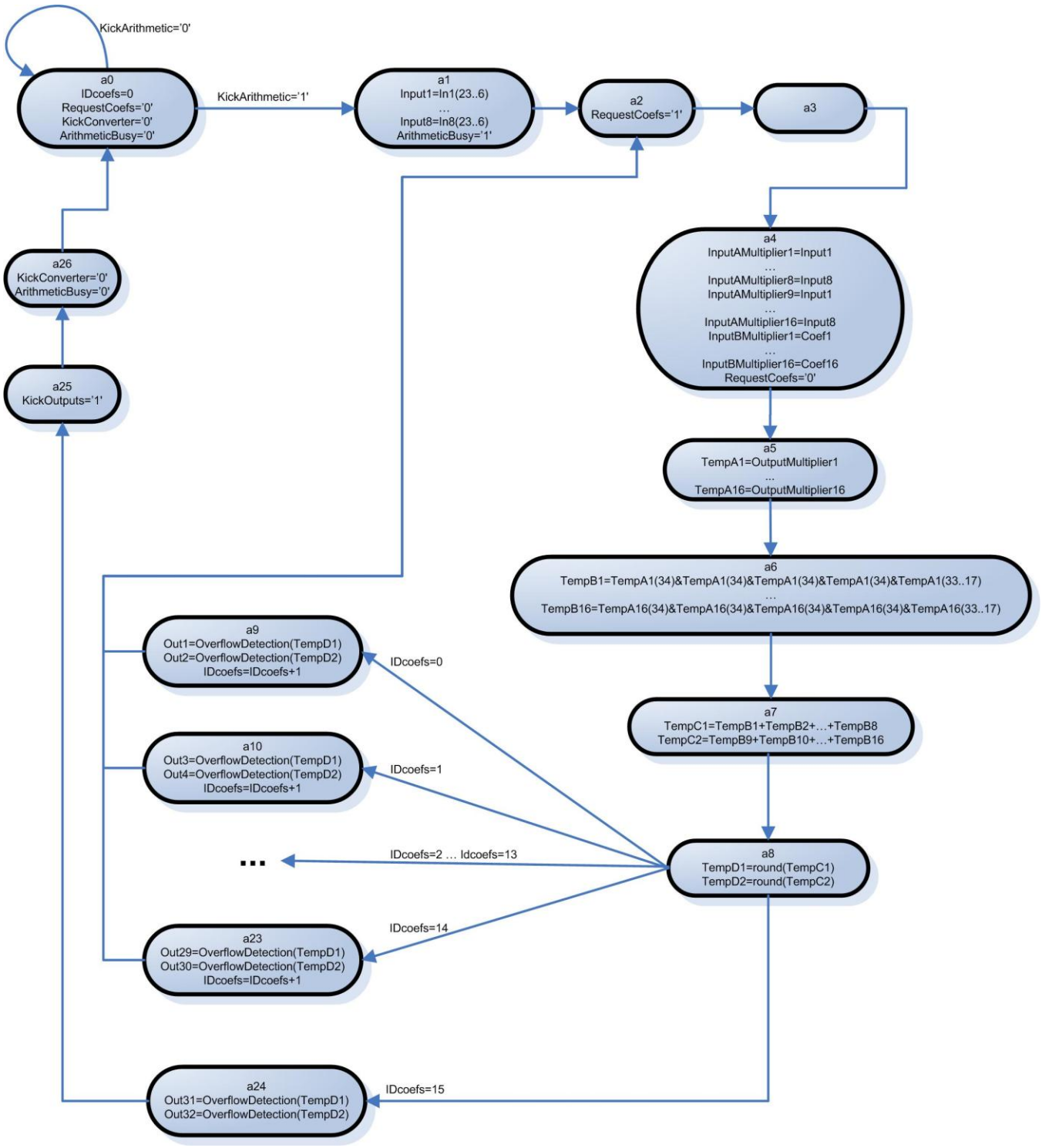


Figura 14 – Máquina de Estados: Aritmético

SINAL	DESCRIÇÃO	TIPO	Número de Bits
KickArithmetic	Sinalização de informação pronta ao bloco Aritmético	Entrada	1
ArithmeticBusy	Sinalização do estado do bloco Aritmético	Saída	1
IDcoefs	Identificação de quais os coeficientes pretendidos no ciclo em curso	Saída/Interno	4
RequestCoefs	Flag de pedido dos coeficientes	Saída	1
KickConverter	Sinaliza informação pronta ao bloco Conversão de Formato	Saída	1
In	Registo de entrada proveniente do bloco de Entrada	Entrada	24
Input	Registo onde é armazenada a amostra de entrada	Interno	18
InputAMultiplier	Registo de Entrada do multiplicador	Interno	18
InputBMultiplier	Registo de Entrada do multiplicador	Interno	18
OutputMultiplier	Registo de Saída do multiplicador	Interno	36
TempA	Registo de Armazenamento do resultado da multiplicação	Interno	36
TempB	Registo de Preparação para o Somador	Interno	21
TempC	Registo de Armazenamento da Soma	Interno	21
TempD	Registo de Armazenamento do Algoritmo de Arredondamento	Interno	19
Out	Registo de Saída	Saída	16

Tabela 3 - Descrição dos Sinais: Bloco Aritmético

Cada ciclo do bloco Aritmético gera duas saídas. São recolhidos os oito sinais de entrada e dezasseis coeficientes, efectuam-se os produtos entre eles, somam-se os resultados respectivos e sujeita-se o resultado a arredondamento e detecção de overflow para que seja entregue ao bloco seguinte. Na Figura 15 assim como na Figura 16 na é possível observar-se de uma forma mais detalhada a evolução de cada sinal. Em cada um dos dezasseis ciclos são usados dois blocos idênticos ao representado na Figura 15. Como se pode ver, o sinal de entrada que é representado em 24bits é reduzido a 18 bits uma vez que esse é o tamanho dos registos de

entrada do multiplicador. Por sua vez, como os coeficientes da matriz são representados em 16 bits, são concatenados dois zeros no lado menos significativo. Estas duas palavras são introduzidas no multiplicador que gera um resultado com o dobro dos bits, ou seja, 36bits. Para serem somados os oito produtos, é feita uma redução de bits desprezando os 17 bits menos significativos e uma expansão do sinal, ou seja, são concatenados dois bits iguais ao bit mais significativo da palavra. Resultam então quatro bits de sinal e dezassete de dados. Na fase seguinte é feito o arredondamento das somas e a detecção de overflow. Estes algoritmos são explicados nos tópicos seguintes.

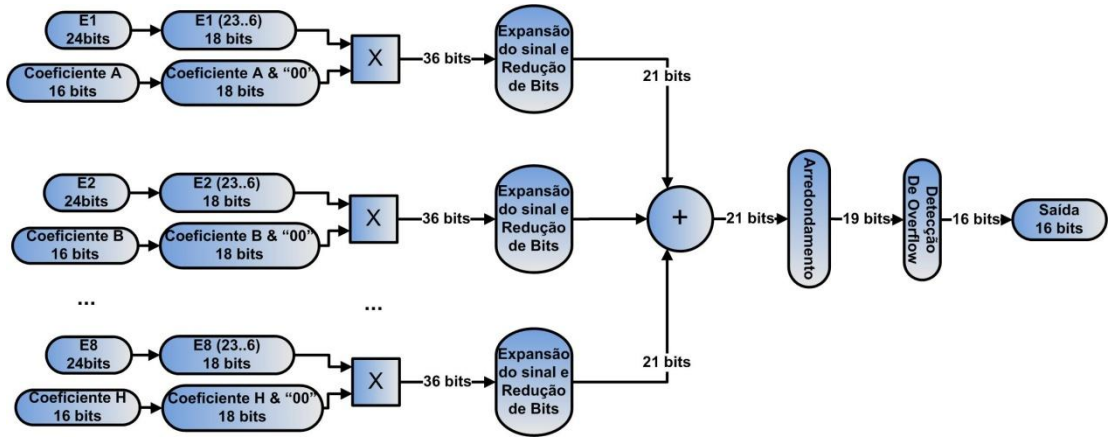


Figura 15 - Evolução do Sinal: Diagrama de Blocos

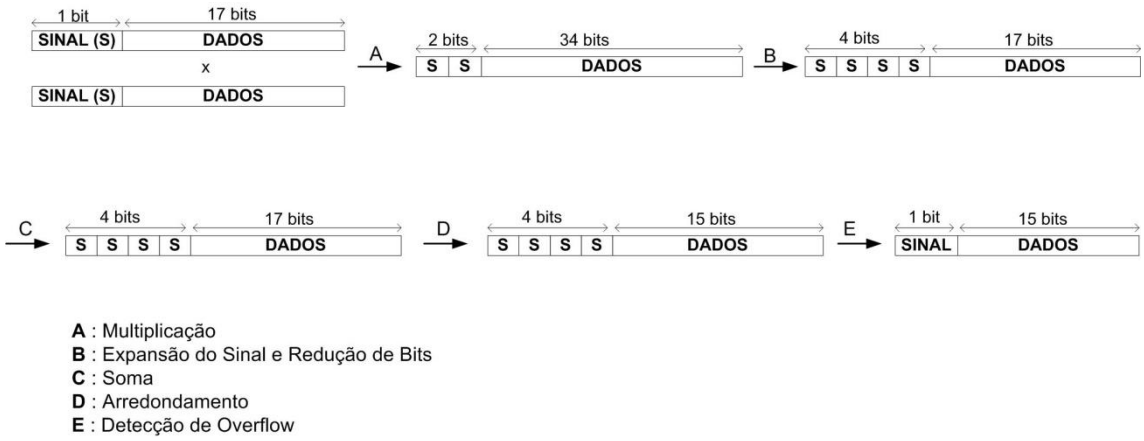


Figura 16 - Evolução do Sinal: Representação da Trama

Algoritmo de Arredondamento

Tendo como entrada uma palavra de 21 bits em complemento para dois (o bit mais significativo tem como índice o número vinte e o menos significativo o número zero), este algoritmo, Figura 17, tem como objectivo reduzir o número de bits para 19. Assim sendo, são avaliados os dois bits menos significativos e, se estes forem "00" ou "01", os dezanove bits mais significativos são mantidos e os restantes dois descartados. No caso de serem "10" ou "11" é somado ou subtraído "1" ao terceiro bit menos significativo no caso de a palavra ser positiva ou negativa, respectivamente. Para esta decisão, avalia-se o bit mais significativo. Os dois bits menos significativos são igualmente descartados.

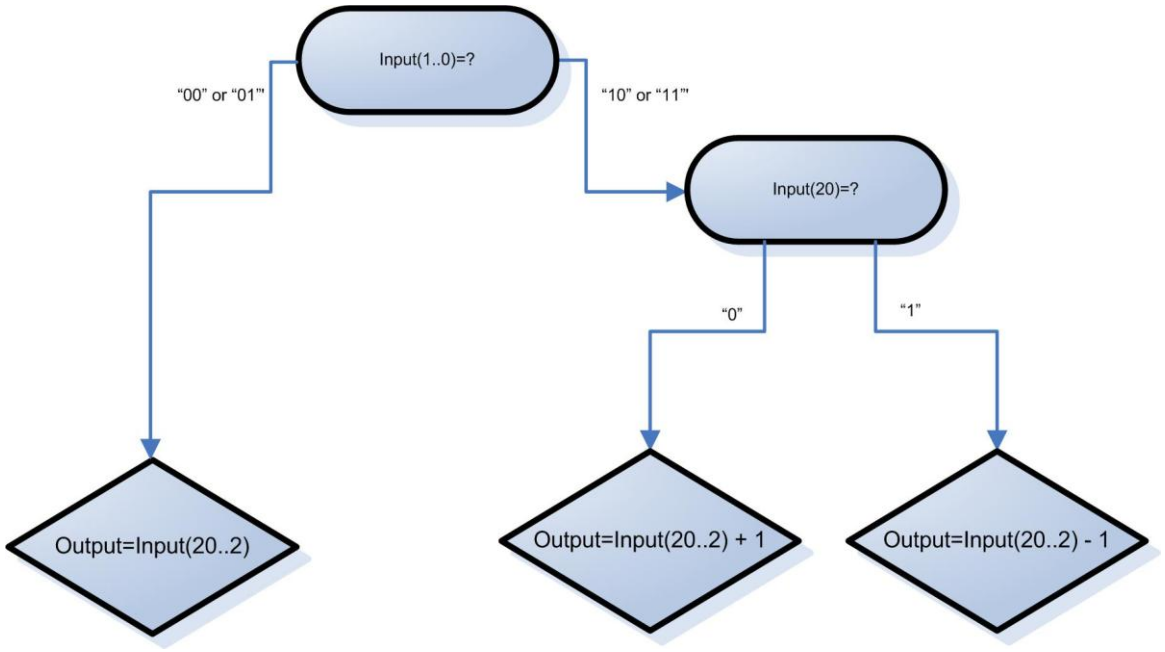


Figura 17 - Algoritmo de Arredondamento

Algoritmo de Detecção de Overflow

O algoritmo de detecção de overflow é usado para detectar se ocorreu overflow na soma dos oito sinais após o produto das entradas com os respectivos coeficientes. O bit mais significativo da palavra de 19bits, cujo índice é dezoito, é avaliado e, consoante o resultado, é feita uma comparação com o maior valor positivo ou negativo dependendo do caso. Se a palavra de entrada for maior que o valor limite, o resultado é saturado. Desta forma, consegue garantir-se que as somas não criam um resultado inválido. Realça-se aqui o facto de se ter feito uma expansão do sinal antes de ter sido realizada a soma. Como ilustra o diagrama da Figura 18, os quatro bits mais significativos são todos de sinal. No final, três dos bits mais significativos são

retirados e os restantes mantidos ou igualados a um dos valores limite se tiver ocorrido overflow.

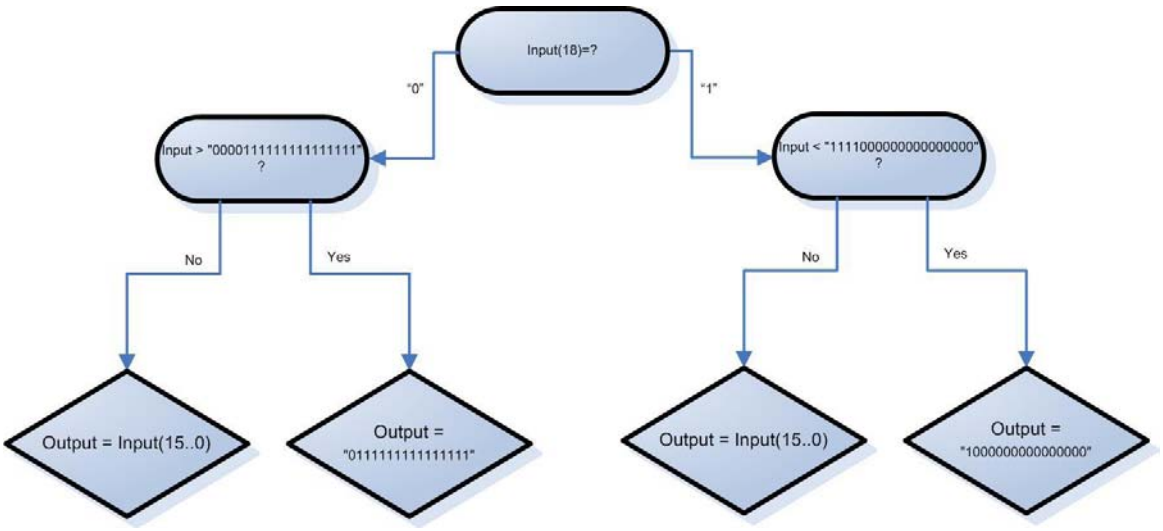


Figura 18 - Algoritmo de Detecção de Overflow

Exercício Exemplificativo

A Figura 19 representa um exemplo da evolução do sinal. Neste caso temos duas palavras de 5bits representados em complemento para dois. Estes dois registos são multiplicados entre si gerando assim um resultado com 10bits em que os dois mais significativos são de sinal. Neste exemplo a soma foi feita com valores nulos, logo, o resultado manteve-se inalterado não havendo portanto overflow. Na fase D ocorreu um arredondamento uma vez que os dois bits mais significativos dos quatro menos significativos descartados ("1000") eram "10". Observando a palavra resultante do arredondamento (1111 1010) verifica-se que os quatros bits mais significativos são todos idênticos, logo, conclui-se que não ocorreu overflow. Porém o método de detecção consiste em comparar este valor com "11110000", i.e., como o bit mais significativo é "1", sendo um resultado negativo, a palavra não pode ser menor que "1111000". Se o primeiro bit fosse "0", o resultado não poderia ser superior a "00001111".

A necessidade de expandir os bits antes da soma surge uma vez que serão somados oito registos diferentes. No pior cenário, o resultado terá mais 3bits de comprimento. Se o bit de sinal for expandido desta forma, os três bits em "excesso" serão representados. Assim sendo, os quatro bits de sinal poderão ser avaliados para detectar se ocorreu overflow na soma dos oito registos.

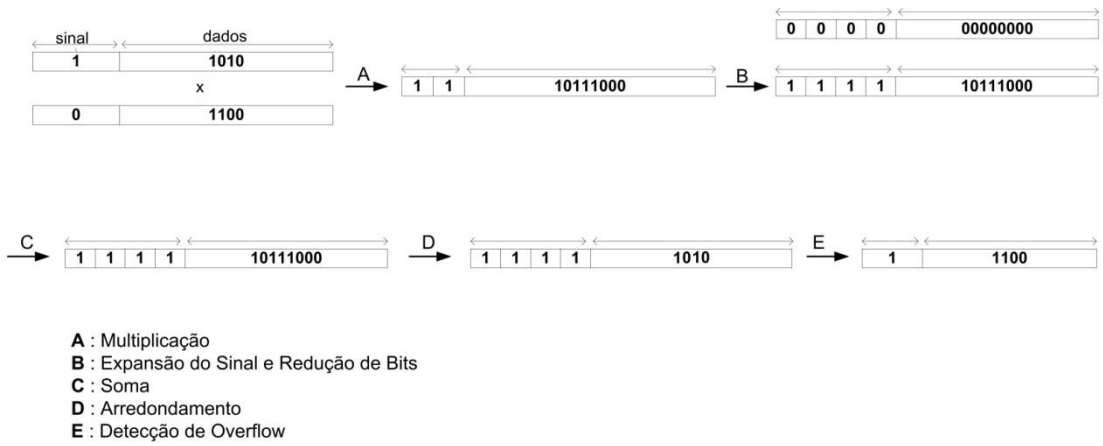


Figura 19 - Exemplo da Evolução do Sinal

3.3.3 – Conversão de Formato: de Complemento para Dois para Binário Natural

Este bloco é incontestavelmente o mais simples. Poderia ter sido integrado no bloco Aritmético mas optou-se por manter separado para destacar o facto de esta conversão ser necessária. Esta separação pode também ser conveniente quando são necessários testes na FPGA em que se deseja retirar o bloco aritmético do processo. Assim sendo, os formatos dos conversores analógico-digital e digital-analógico continuam a ser respeitados. A Figura 20 representa a máquina de estados implementada. O sinal *KickConverter*, proveniente do bloco Aritmético, serve para desencadear o funcionamento desta máquina. Assim que este sinal assume o valor lógico '1', a máquina passa do estado *a0* em que as duas atribuições observadas na figura são feitas e inicia o ciclo representado. No estado *a2* é onde efectivamente ocorre a conversão. Negando o bit mais significativo do sinal de entrada, equivalente a somar o número mais negativo (em complemento para dois), o sinal passa de uma representação entre -2^{-n-1} e $2^{n-1}-1$ em complemento para dois para uma gama entre 0 e 2^n-1 mantendo a relação entre todos os elementos. No estado *a3* existe um compasso de espera semelhante ao do bloco de Entrada para que a sinalização do bloco de saída não seja feita enquanto o mesmo estiver ocupado. A Tabela 4 representa, como no caso anterior, a descrição dos sinais da máquina de estados.

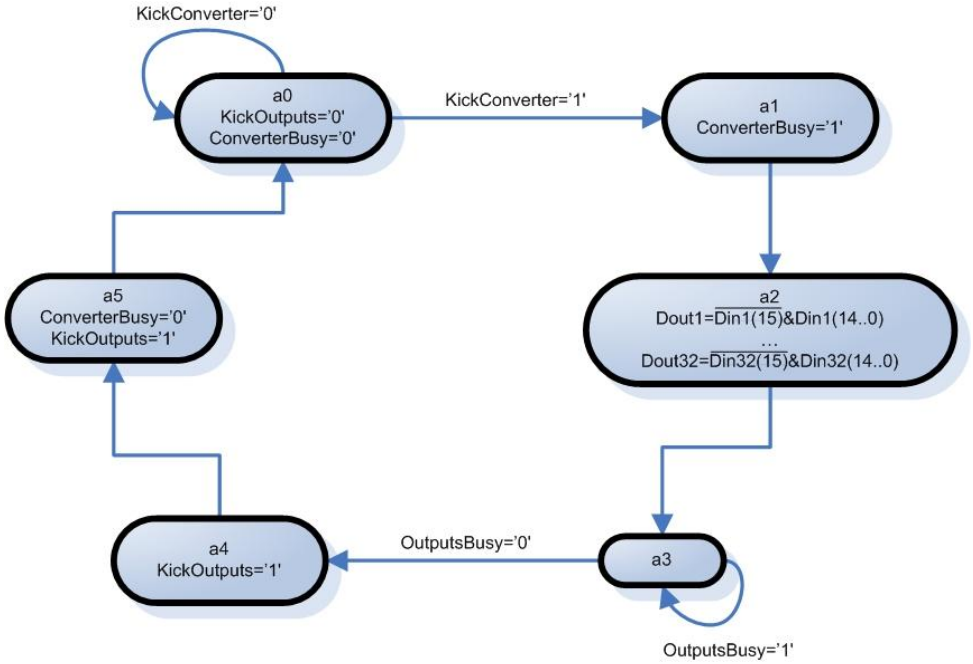


Figura 20 – Máquina de Estados: Conversão de Formatos

SINAL	DESCRIÇÃO	TIPO	Número de Bits
KickConverter	Sinal que sinaliza informação pronta ao bloco Conversão de Formatos	Entrada	1
KickOutputs	Sinal que sinaliza informação pronta ao bloco de Saída	Saída	1
ConverterBusy	Sinalização Interna do estado do Bloco de Conversão	Interno	1
OutputBusy	Sinalização do estado do bloco de Saída	Entrada	1
Din1…Din32	Sinal de Entrada (Complemento para 2)	Entrada	16
Dout1…Dout32	Sinal de Saída (Binário Natural)	Saída	16

Tabela 4 - Descrição dos Sinais: Conversão de Formatos

3.3.4 – Envio das Saídas: Interface com os DACs

O Bloco de Saída, assim como o de Bloco de Entrada e o Bloco de Comunicação USB, é um bloco que interage com o exterior da FPGA. A máquina de estados da Figura 23, cujos sinais são descritos na Tabela 5, tem a responsabilidade de gerar os sinais de controlo dos DACs que permitem a conversão das trinta e duas saídas de digital para analógico. Os sinais a serem gerados pela FPGA estão representados no diagrama temporal da Figura 21 e garantem o funcionamento correcto dos DACs.

O sinal *SCLK* serve de relógio de transmissão de dados enquanto que *SYNK* representa o intervalo de tempo em que está a ser enviada informação. Este sinal deverá estar no nível lógico '0' enquanto se transmitem dados. A linha *DIN* é a linha série onde é colocada a informação digital correspondente aos quatro canais que cada DAC tem de converter. Esta linha é gerada com o auxílio a uma variável, *OutAux*, onde os quatro sinais de saídas que são enviados para cada DAC são concatenados, no estado *a1*, com constantes de configuração dos conversores (descritos na secção 4.3.2 – Conversor Digital-Analógico – DAC8534) assim como alguns bits que serão desprezados pelo conversor. Origina-se assim um registo de 101 bits que será percorrido à medida que se colocam bits na linha série. A variável *Indice* é usada para percorrer todos os índices do registo *OutAux* e para controlar o final do envio de cada canal. O sinal *OnOff* é um sinal que auxilia a activação e desactivação do sinal *SYNK* que sinaliza estado do envio de informação (activo/desactivo). Este envio é simultaneamente feito para todos os DACs.

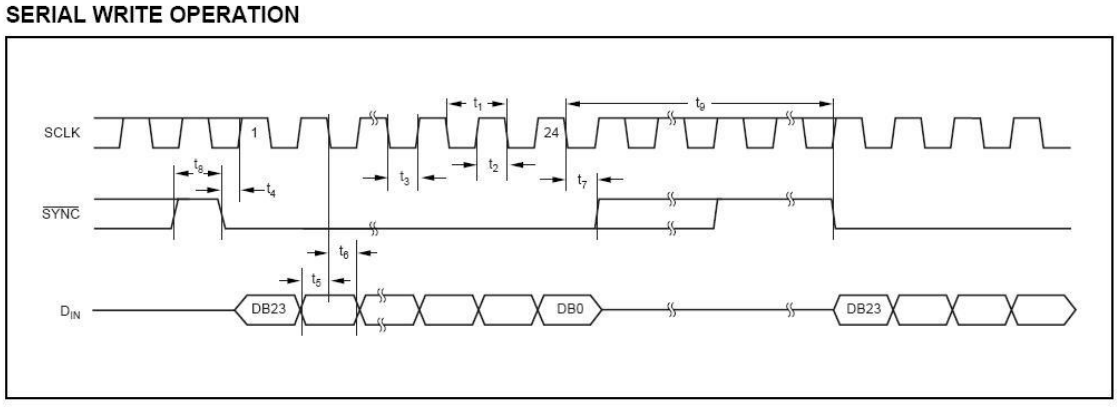


Figura 21 - Diagrama Temporal de Funcionamento do DAC

A máquina de estados fica, como nas descritas anteriormente, no primeiro estado, *a0*, até que seja sinalizada, neste caso, através do sinal *KickOutputs*. Quando esta máquina for

iniciada, procede-se à concatenação dos sinais de saída nas respectivas variáveis e activa-se, no estado seguinte, o sinal *OnOff*. Uma vez que esta máquina é sensível à transição ascendente do relógio e que o processo que gera o sinal *SYNK* é sensível à transição descendente, *SYNK* assume o valor lógico '0' na transição descendente e o primeiro bit é colocado na linha na próxima transição ascendente. Este instante pode ser identificado na Figura 22 como *A*. Este diagrama representa o espaço de tempo entre o envio de dois canais, incluindo assim o momento de finalização do envio de um canal e de activação do canal seguinte. Esta implementação foi feita desta forma para que, se houver um atraso do sinal *SYNK* em relação ao de dados, não seja perdido nenhum bit. Este acontecimento resultaria num erro de configuração e conversão. O sinal *OnOff* é mantido activo e em cada ciclo de relógio é colocado o bit seguinte na linha série. Quando o sinal *Indice* indicar o fim do envio do canal em questão, o envio termina. Após o envio de cada canal é feita uma espera que permite respeitar a pausa entre envios imposta pelas especificações dos DACs. Esta pausa é criada com auxílio à variável *Counter*, como é possível observar no estado *a5* assim como no diagrama da Figura 22. Após o envio do quarto canal para cada DAC, a máquina regressa, não ao estado *a3* (onde seria enviado o próximo canal) mas sim, ao estado *a0* onde é feita a reinicialização das variáveis de controlo da máquina.

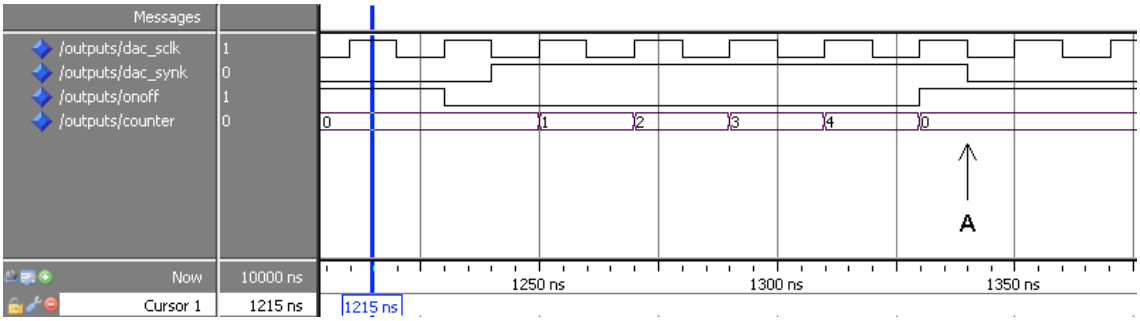


Figura 22 - Geração de Sinais para Controlo do DAC

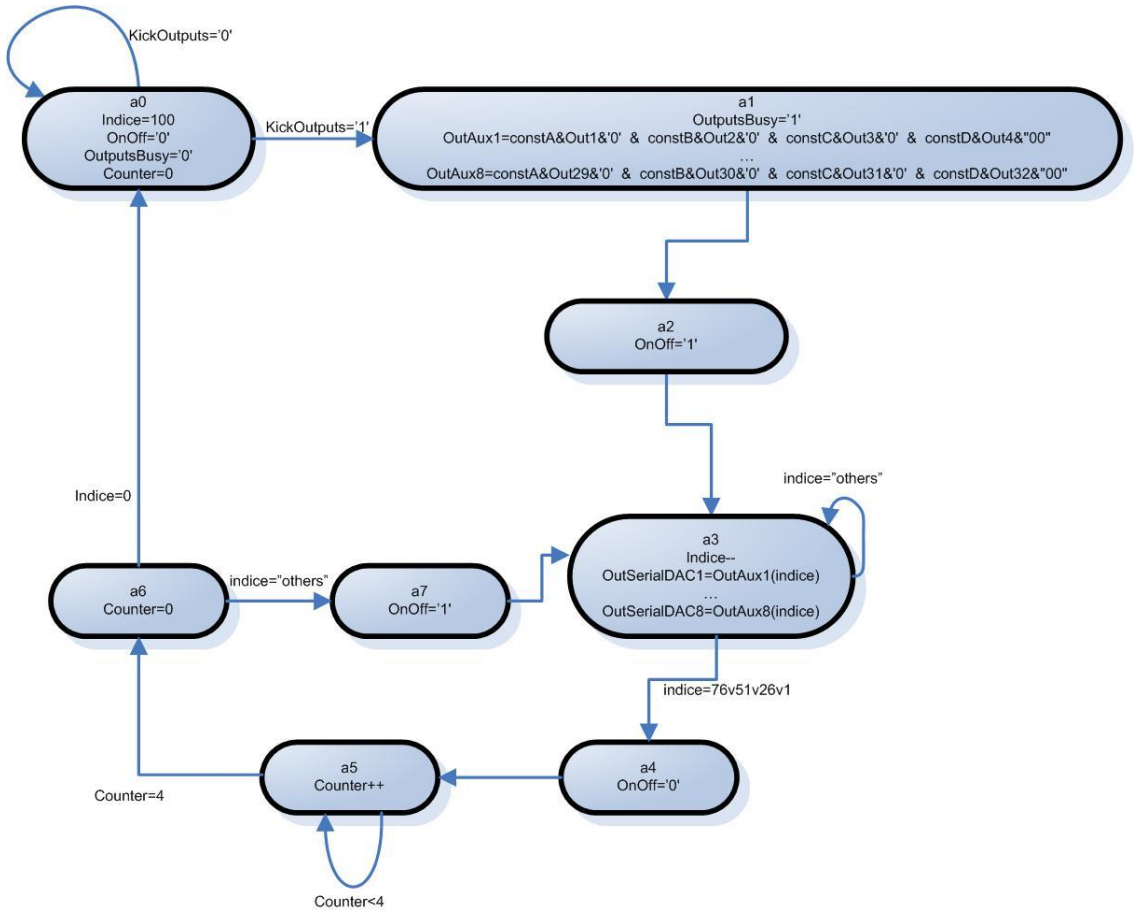


Figura 23 – Máquina de Estados: Saída

SINAL	DESCRIÇÃO	TIPO	Número de Bits
KickOutputs	Sinalização de informação pronta ao bloco de Saída	Saída	1
OutputBusy	Sinalização do estado do bloco de Saída	Entrada	1
Indice	Identificação do Índice actual da variável OutAux	Interno	Inteiro
Counter	Contador para Paus entre envio de canais	Interno	3
OnOff	Sinal auxiliar para controlo de SYNK	Interno	1
Const	Registo Auxiliar para a configuração dos ADCs	Interno	8
Out	Registo de Entrada	Entrada	16
OutAux	Registo Auxiliar para Envio para DACs	Interno	101
OutSerialDAC	Linha Série de Dados para DACs	Saída	1

Tabela 5 - Descrição dos Sinais: Bloco de Saída

3.3.4 – Controlo da Memória: Criação e Acesso aos Blocos de Memória Embutida

Os bancos de memória embutidos (Block RAMs) na FPGA usados neste projecto são, como se poderá observar no Capítulo 4, bancos de memória *dual-port* com 16384bits de armazenamento. Cada porto pode ter 512 posições de memória com 32bits de comprimento cada. O acesso a cada um dos dois portos pode ser feito em simultâneo permitindo assim a leitura de 64bits (32 de cada porto) apenas num ciclo de relógio.

No bloco Aritmético verificou-se que seria necessário recolher dezasseis coeficientes de 16bits de cada vez para que, em cada ciclo, sejam feitos dezasseis produtos de forma a gerar duas saídas. Uma vez que se pretende que essa leitura seja simultânea, usaram-se quatro Block RAMs com estas características para que se possam ler oito registos de 32bits de uma só vez. Em cada endereço de memória devem então guardar-se dois coeficientes. Uma vez que existem 256 coeficientes, cada banco de memória armazenará sessenta e quatro coeficientes, logo, terão ocupados 32 das suas 512 posições de memória. O espaço inutilizado em cada banco é uma contra-partida que surge pela necessidade de acesso simultâneo a 16 coeficientes de forma a optimizar o bloco aritmético.

O bloco de Controlo de Memória, sempre que é feita uma escrita através da comunicação USB, recebe um sinal de pedido de armazenamento por parte do bloco que estabelece a comunicação em causa. Existe então um processo dentro do bloco de gestão de memória, responsável por determinar constantemente qual a posição de memória e o respectivo banco que corresponde ao endereço da comunicação USB da informação a ser enviada. Esta descodificação é feita avaliando o valor do barramento de endereço da lógica USB. O outro processo existente usa a informação da descodificação de endereços assim como a linha que sinaliza o pedido de armazenamento, para armazenar todos os registos enviados. É de notar que, como em cada posição de memória dos Block RAMs, existem dois coeficientes armazenados, um novo armazenamento de um registo requer que o outro coeficiente presente na mesma posição de memória do banco seja salvaguardado. Para isto, é lida a palavra de 32bits do endereço em causa que corresponde aos dois coeficientes naquela posição de memória. Substituem-se os menos ou mais 16 bits significativos, dependendo do coeficiente a actualizar, e armazenam-se de novo os 32bits.

Na Tabela 6 é possível observar a posição de cada coeficiente na memória dos quatro bancos usados. IXCY representa o coeficiente correspondente à entrada número X na coluna Y. Observa-se então que cada banco de memória tem trinta e duas posições ocupadas tendo cada uma delas dois coeficientes da matriz armazenados. Acima de cada célula é possível observar o endereço USB que permitirá registar o coeficiente desejado. Como já foi referido, o bloco

aritmético gera duas saídas de cada vez, logo, o acesso à memória é feito de forma a serem lidas duas posições de memória de cada Block RAM. Esta tabela deve ser consultada para que o envio a partir do computador que gere a matriz de mistura de áudio seja feito de forma a garantir o comportamento desejado.

Foi também implementada neste bloco a possibilidade de leitura dos coeficientes de modo a serem observados nos LEDs existentes na placa de desenvolvimento que contém a FPGA, NEXYS2. Uma vez que cada coeficiente tem 16bits e que existem apenas 8 leds, é usado um interruptor que define se são mostrados os bits mais ou menos significativos. Esta funcionalidade é usada para depuração e poderá numa versão final ser retirada.

A lógica de controlo deste bloco foi implementada de forma a garantir que nunca nenhum envio USB é falhado uma vez que isso teria um impacto grave no funcionamento do sistema. Por outro lado, se ocorrer uma escrita em simultâneo com um pedido por parte do Bloco Aritmético o pior cenário possível consiste no erro da amostra referente aos canais que estavam nessa altura a ser processados.

DISTRIBUIÇÃO DOS COEFICIENTES NA MEMÓRIA

posição de memória	Banco I (0-63)		Banco II (64-127)		Banco III (128-191)		Banco IV (192-255)	
	endereço USB: 0..3		endereço USB: 64..67		endereço USB: 128..131		endereço USB: 192..195	
0	I1C1	I2C1	I5C1	I6C1	I1C2	I2C2	I5C2	I6C2
1	I3C1	I4C1	I7C1	I8C1	I3C2	I4C2	I7C2	I8C2
	endereço USB: 4..7		endereço USB: 68..71		endereço USB: 132..135		endereço USB: 196..199	
2	I1C3	I2C3	I5C3	I6C3	I1C4	I2C4	I5C4	I6C4
3	I3C3	I4C3	I7C3	I8C3	I3C4	I4C4	I7C4	I8C4
	endereço USB: 8..11		endereço USB: 72..75		endereço USB: 136..139		endereço USB: 200..203	
4	I1C5	I2C5	I5C5	I6C5	I1C6	I2C6	I5C6	I6C6
5	I3C5	I4C5	I7C5	I8C5	I3C6	I4C6	I7C6	I8C6
	endereço USB: 12..15		endereço USB: 76..79		endereço USB: 140..143		endereço USB: 204..207	
6	I1C7	I2C7	I5C7	I6C7	I1C8	I2C8	I5C8	I6C8
7	I3C7	I4C7	I7C7	I8C7	I3C8	I4C8	I7C8	I8C8
	endereço USB: 16..19		endereço USB: 80..83		endereço USB: 144..147		endereço USB: 208..211	
8	I1C9	I2C9	I5C9	I6C9	I1C10	I2C10	I5C10	I6C10
9	I3C9	I4C9	I7C9	I8C9	I3C10	I4C10	I7C10	I8C10
	endereço USB: 20..23		endereço USB: 84..87		endereço USB: 148..151		endereço USB: 212..215	
10	I1C11	I2C11	I5C11	I6C11	I1C12	I2C12	I5C12	I6C12
11	I3C11	I4C11	I7C11	I8C11	I3C12	I4C12	I7C12	I8C12
	endereço USB: 24..27		endereço USB: 88..91		endereço USB: 152..155		endereço USB: 216..219	
12	I1C13	I2C13	I5C13	I6C13	I1C14	I2C14	I5C14	I6C14
13	I3C13	I4C13	I7C13	I8C13	I3C14	I4C14	I7C14	I8C14
	endereço USB: 28..31		endereço USB: 92..95		endereço USB: 156..159		endereço USB: 220..223	
14	I1C15	I2C15	I5C15	I6C15	I1C16	I2C16	I5C16	I6C16

15	I3C15	I4C15	I7C15	I8C15	I3C16	I4C16	I7C16	I8C16
	endereço USB: 32..35		endereço USB: 96..99		endereço USB: 160..163		endereço USB:224..227	
16	I1C17	I2C17	I5C17	I6C17	I1C18	I2C18	I5C18	I6C18
17	I3C17	I4C17	I7C17	I8C17	I3C18	I4C18	I7C18	I8C18
	endereço USB: 36..39		endereço USB: 100..103		endereço USB: 164..167		endereço USB: 228..231	
18	I1C19	I2C19	I5C19	I6C19	I1C20	I2C20	I5C20	I6C20
19	I3C19	I4C19	I7C19	I8C19	I3C20	I4C20	I7C20	I8C20
	endereço USB: 40..43		endereço USB: 104..107		endereço USB: 168..171		endereço USB: 232..235	
20	I1C21	I2C21	I5C21	I6C21	I1C22	I2C22	I5C22	I6C22
21	I3C21	I4C21	I7C21	I8C21	I3C22	I4C22	I7C22	I8C22
	endereço USB: 44..47		endereço USB: 108..111		endereço USB: 172..175		endereço USB: 236..239	
22	I1C23	I2C23	I5C23	I6C23	I1C24	I2C24	I5C24	I6C24
23	I3C23	I4C23	I7C23	I8C23	I3C24	I4C24	I7C24	I8C24
	endereço USB: 48..51		endereço USB: 112..115		endereço USB: 176..179		endereço USB: 240..243	
24	I1C25	I2C25	I5C25	I6C25	I1C26	I2C26	I5C26	I6C26
25	I3C25	I4C25	I7C25	I8C25	I3C26	I4C26	I7C26	I8C26
	endereço USB: 52..55		endereço USB: 116..119		endereço USB: 180..183		endereço USB: 244..247	
26	I1C27	I2C27	I5C27	I6C27	I1C28	I2C28	I5C28	I6C28
27	I3C27	I4C27	I7C27	I8C27	I3C28	I4C28	I7C28	I8C28
	endereço USB: 56..59		endereço USB: 120..123		endereço USB: 184..187		endereço USB:248..251	
28	I1C29	I2C29	I5C29	I6C29	I1C30	I2C30	I5C30	I6C30
29	I3C29	I4C29	I7C29	I8C29	I3C30	I4C30	I7C30	I8C30
	endereço USB: 60..63		endereço USB: 124..127		endereço USB: 188..191		endereço USB: 252..255	
30	I1C31	I2C31	I5C31	I6C31	I1C32	I2C32	I5C32	I6C32
31	I3C31	I4C31	I7C31	I8C31	I3C32	I4C32	I7C32	I8C32

Tabela 6 - Distribuição dos Coeficientes na Memória

3.3.5 – Controlo USB: Digilent Parallel Interface Module

Este bloco surgiu de uma adaptação de um projecto providenciado pela fabricante da NEXYS2, *Digilent®*, e é denominado como *Digilent® Parallel Interface Module Reference* Design [8]. A lógica inicial permitia o envio via USB de oito palavras de 8bits cada cujo armazenamento seria feito em registos. Para este projecto, dadas as características e requisitos pretendidos, foi necessária uma adaptação que permitisse o envio de duzentas e cinquenta e seis palavras de 16bits. O respectivo armazenamento em registos resultaria num desperdício de recursos, logo, foram introduzidos os bancos de memória já referidos onde é guardada esta informação. A máquina de estados implementada permite a configuração da FPGA de forma a que respeite os diagramas representados na Figura 24 [8]. Os dois barramentos de 8bits existentes são usados para colocar o endereço e os dados, respectivamente, uma linha que define

se a operação em causa é de leitura ou escrita, outras duas que sinalizam a activação da operação de envio/recepção de endereço ou dados e uma última de espera, sendo denominadas como *WRITE*, *ASTB* (address strobe), *DSTB* (data strobe) e *WAIT*, respectivamente. Antes de ser enviada informação é necessário o envio do endereço dos dados correspondentes. Dado que os coeficientes que são necessários são de 16bits e que o barramento de dados é apenas de oito, é necessário repartir o envio em dois. Enviam-se então os oito bits menos significativos seguidos dos mais significativos. Apenas quando estes últimos chegam é feito um pedido de armazenamento ao bloco que faz a gestão dos bancos de memória para que o coeficiente enviado seja gravado num dos bancos utilizados.

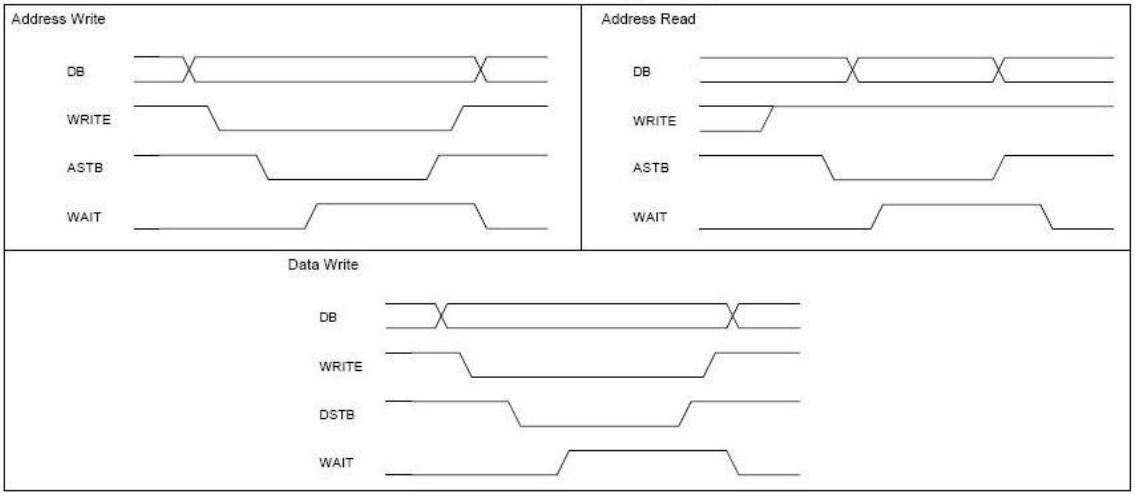


Figura 24 - Diagrama Temporal: Comunicação USB [8]

Capítulo 4 – SELECÇÃO E APRESENTAÇÃO DO HARDWARE

Para criar um sistema apto para receber sinais áudio analógicos de forma a manter a sua integridade e possibilitar que estes sejam processados digitalmente, foi necessário reunir hardware cuidadosamente de forma a garantir a qualidade do sinal.

O processamento central de todo o sistema é realizado numa *Field Programmable Gate Array* (FPGA). Assim sendo, foi escolhida uma placa de desenvolvimento da Digilent® que contém uma FPGA da família Spartan3E juntamente com interfaces adicionais que podem ser aproveitados. Como os sinais analógicos à entrada são processados digitalmente, são necessários conversores analógico-digital e digital-analógico para converter os sinais de novo para formato analógico.

Para adaptar os sinais externos ao sistema foram colocados *buffers* à entrada e à saída e também dimensionados filtros para eliminar informação que possa existir em frequências fora da gama de interesse. Neste capítulo será também abordada a forma como o sistema é alimentado.

4.1 – Placa de desenvolvimento – NEXYS2 de Digilent®

A NEXYS2 é uma placa desenvolvida pela *Digilent*® que tem como núcleo uma FPGA da *Xilinx*®, a Spartan3E-500 FG320 [18]. Esta placa tem vários interfaces ilustrados na Figura 25 [6], que permitem uma interacção directa com a FPGA. Entre eles destacam-se:

- Comunicação USB2.0
- Plataforma Flash para configuração não-volátil da FPGA
- Cristal de 50MHz
- Memória Flash
- Memória SDRAM
- Conector de Alta Velocidade (Hirose FX-2)
- 4 Conectores PMOD de 12 pinos
- Comunicação RS232
- Conector VGA
- Conector PS/2

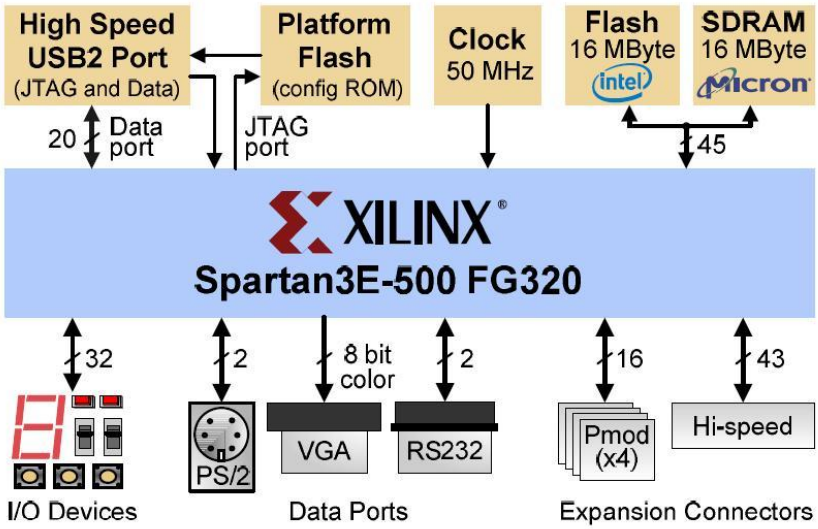


Figura 25 - Esquemático da NEXYS2 [6]

Ligados à FPGA existem também outros periféricos como oito *Light Emitting Diods* (LEDs), quatro *displays* de sete segmentos, oito interruptores e quatro botões de pressão cuja funcionalidade pode ser livremente reconfigurada. A comunicação USB2.0 é usada, como já se verificou, para transmitir a informação que define a matriz de mistura enquanto que os conectores existentes são usados para ligar os conversores analógico-digital assim como os digital-analógico.

O cristal de 50MHz embutido na placa é usado para criar o sinal de relógio que controla grande parte da lógica da FPGA.

4.1.1 – Comunicação USB – Digilent Adept SDK

Dada a constante necessidade de comunicação entre vários sistemas, a *Digilent*® criou e disponibilizou um protocolo que permite a comunicação entre os seus produtos e aplicações a correrem num computador com o Microsoft Windows® denominado como *Digilent Port Communications Utility* (DPCUTIL) [7]. No caso da NEXYS2, no que respeita à comunicação USB, este conjunto de bibliotecas e funções comunicam com o módulo *Cypress CY7C68013* presente na placa (Figura 26 [9]) que converte o formato da informação recebida num aspecto idêntico ao da comunicação paralela apresentada Figura 24. Assim sendo, é necessária a configuração da FPGA para que esta interprete correctamente os sinais provenientes do módulo *Cypress*. Esta configuração foi demonstrada na secção 3.3.5 – Controlo USB: Digilent Parallel Interface Module (Digilent® Parallel Interface Module Reference Design [8]).

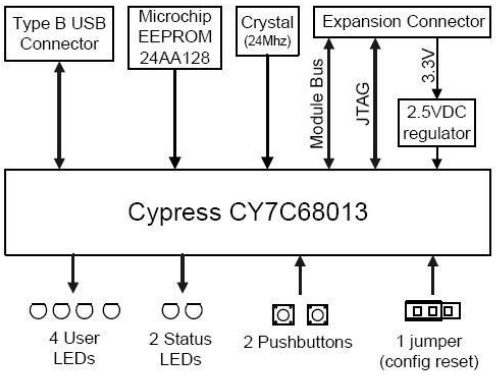


Figura 26 - Módulo Cypress CY7C68013 [9]

Da parte do computador que tem a ligação USB, o algoritmo fornecido pela *Digilent*® que invoca as funções do protocolo DPCUTIL, foi criado em C++ e o respectivo projecto, como recomendado, criado em Visual Studio C++.

Uma vez sintetizado o circuito da FPGA e preparado o projecto que estabelece a comunicação USB, é necessário criar uma *Dynamic Link Library* (DLL) para que este programa possa ser usado a mais alto nível.

Actualmente os parâmetros que definem a matriz de mistura de áudio podem ser alterados ao invocar um executável numa linha de comandos DOS com os seguintes parâmetros:

>> nexys_usb.exe –p 20 255

>> nexys_usb –p 20 128

O coeficiente presente na posição de memória *20* assumirá o valor 0x80ff uma vez que os 16 bits menos significativos são 255 em base decimal e os 16 bits mais significativos são 128 também em base decimal.

4.2 – FPGA Spartan3E-500 FG320 de Xilinx®

A FPGA disponível na placa contém cerca de 500 000 portas de sistema que podem ser reconfiguradas livremente. Integrados na FPGA existem vários blocos de hardware que permitem um funcionamento optimizado dos sistemas desenvolvidos estando alguns dos blocos representados na Figura 27 [18]. Entre eles destacam-se, neste projecto, os vinte multiplicadores dedicados, os vinte bancos de memória embutidos (Block RAMs), que são, neste caso, usados para armazenar a informação enviada por USB, e os quatro *Digital Clock Manager´s* (DCM) que garantem uma sincronização de relógio optimizada entre todo o hardware configurado. Na figura temos também representados os blocos lógicos configuráveis (CLBs) assim como os blocos de entrada e de saída (IOBs).

Os *Digital Clock Managers,* assim como buffers internos (BUFG), têm acesso a linhas dedicadas de relógio presentes na FPGA de forma a minimizar os desfasamentos do sinal de relógio entre os vários pontos do circuito sintetizado. Os DCMs são entidades usadas para gerir sinais de relógio. Entre as várias funcionalidades presentes nestes módulos, é possível minimizar o atraso do sinal entre os vários pontos da lógica sintetizada usando um sinal de feedback como referência, criar um divisor de frequência a partir do sinal original e também fazer mudanças de fase ao sinal. Os DCMs são bastante importantes na medida em que trazem estabilidade ao circuito digital, uma vez que todo ele é gerido por relógios.

Os multiplicadores dedicados são componentes embutidos na FPGA que permitem efectuar o produto entre dois operandos de 18bits em complemento para dois e geram um resultado de 36bits. Esta operação não necessita de lógica adicional e é feita apenas num ciclo de relógio.

Os bancos de memória dedicados usados neste projecto são *dual-port* com possibilidade de acesso síncrono. É possível o endereçamento de 512 posições de memória em cada um dos portos do banco tendo, cada registo, 32bits de comprimento. O funcionamento destes componentes embutidos, assim como uma descrição mais detalhada, podem ser encontrados em [17].

A *Xilinx*® disponibiliza vários elementos denominados como *IP Cores* que podem ser instanciados e usados livremente no projecto. Ferramentas de depuração, processamento de sinal (transformadas, filtros digitais, entre outros), algoritmos matemáticos e outros elementos como contadores, comparadores, *shift-registers* são algumas das opções entre as várias existentes. Todos estes blocos são reutilizáveis e algumas das suas características ajustáveis conforme a necessidade do projectista. Todos eles foram criados de forma a ter um funcionamento óptimo. Estes módulos são denominados como *Xilinx LogiCORE™ IP*. Um

elemento que é usado neste projecto é a criação de filas *First-In-First-Out* (FIFOs) como foi visto na secção sobre a lógica interna da FPGA (3.3 – Lógica Interna da FPGA).

A documentação da FPGA pode ser encontrada em [18]. Em [17] existem indicações sobre como usar os blocos embutidos na FPGA.

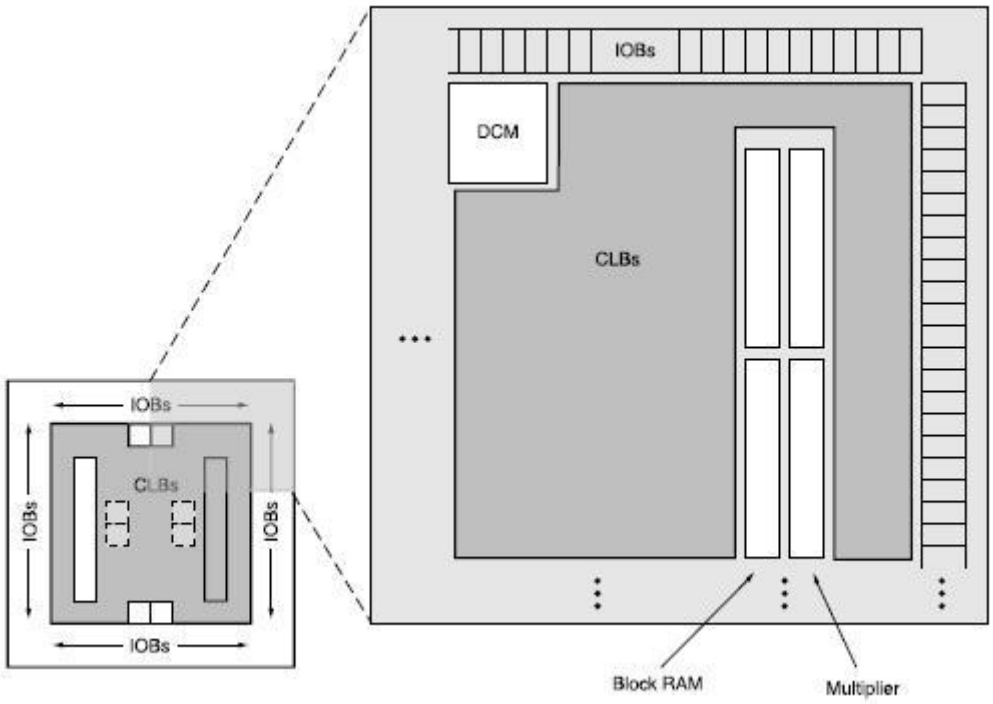


Figura 27 - Diagrama de alguns componentes da FPGA [18]

4.3 – Conversores Analógico-Digital e Digital-Analógico

A necessidade de usar conversores surge na sequência da utilização de uma FPGA para processar sinais analógicos. Assim sendo, os sinais áudio de entrada terão de ser convertidos de analógico para digital para serem processados. Uma vez obtido o sinal de saída na FPGA, este terá de voltar a ser convertido para analógico para ser posteriormente reproduzido.

A escolha dos conversores passa por vários aspectos dando, entre eles, mais importância à qualidade do sinal e à sincronização entre os vários canais. É crucial garantir a qualidade do sinal áudio para que o produto final mantenha a pureza sonora que é fornecida à entrada. Considerando que existem vários sinais à entrada, o desfasamento entre eles terá de ser nulo.

Logo, é necessário que tanto a amostragem dos sinais como o seu processamento seja síncrono. Vão ser apresentadas de seguida as especificações dos conversores que foram escolhidos de forma a cumprir estes requisitos. O PCM1802 foi seleccionado para fazer a conversão para digital e o DAC8534 para a conversão analógica. Os respectivos *datasheets* são [3] e [5] e o seu funcionamento e interacção com a FPGA são brevemente descritos nas próximas duas secções.

4.3.1 – Conversor Analógico-Digital – PCM1802

Descrição do Componente

O PCM1802 [3] é um conversor analógico-digital da *Burr-Brown Products (Texas Instruments).* Este conversor é habitualmente usado como um conversor *stereo* mas, neste caso, as duas entradas serão encaradas como dois canais distintos. Dois sinais áudio entram por cada uma das entradas e a respectiva conversão sai por apenas uma linha digital. Na Figura 28 observam-se a entrada dos dois canais analógicos no ADC e a saída das linhas que controlam o envio da informação digital. O sinal *DOUT* é a linha onde são colocados os dados enquanto que os restantes sinais são usados para estabelecer a comunicação com o elemento a que o conversor está ligado. Os bits que compõem a palavra binária que caracteriza o sinal são enviados via série. Este conversor apresenta uma resolução de 24 bits (representação em complemento para 2) e usa dois moduladores *Sigma-Delta* para realizar a conversão dos dois canais. Para garantir uma largura de banda que abrange apenas a gama áudio, este integrado tem à entrada um filtro anti-aliasing e um filtro passa-alto que remove, se assim o for desejado, a componente DC.

Tendo como modos de funcionamento, *Master e Slave*, o conversor foi configurado de forma a se comportar como *Master* para que seja independente no que respeita à amostragem e ao envio da informação. Este componente necessita de um oscilador externo que neste caso lhe fornece uma frequência de relógio de 24,576MHz [10] ($256xf_a$, em que f_a é a frequência de amostragem). A frequência de amostragem resultante é de 96KHz, sendo este o máximo possível para este conversor.

Para se receberem oito canais áudio de entrada é necessário usar quatro conversores PCM1802 uma vez que cada um dispõe de dois canais. Estes componentes partilham o oscilador e alguns sinais de configuração provenientes da FPGA. O esquema de ligações é apresentado no anexo (Anexo E – Placas de Circuito Impresso).

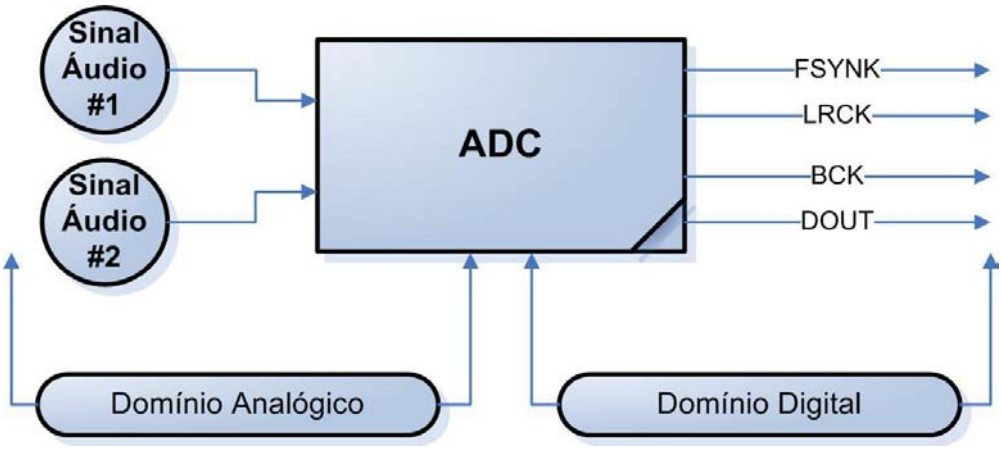


Figura 28 - Conversor Analógico-Digital

Sincronização ADC/FPGA

Sendo necessária uma sincronização com a FPGA, os pinos *BCK*, *LRCK* e *FSYNK* representados na saída do ADC na Figura 28 e no diagrama da Figura 29 são os necessários para tal efeito. Entre algumas escolhas possíveis, a análise descrita de seguida foi a escolhida como protocolo de comunicação entre os dois elementos. O sinal *FSYNK* sinaliza o envio ou paragem de informação enquanto *LRCK* indica qual dos canais se encontra a ser transmitido. O sinal *BCK* serve de relógio para o *shift-register*. Os sinais *FSYNK* e *LRCK*, ao sofrerem uma transição ascendente simultânea, marcam o início da transferência dos dados referente ao canal esquerdo, ou canal 1. Cada ciclo do sinal *BCK* representa a passagem de um bit da amostra, desde o mais significativo ao menos significativo. Os primeiros 24 bits são então transmitidos nos primeiros 24 ciclos de *BCK*. Quando o último bit for enviado, o *FSYNK* retorna ao valor lógico *zero* voltando apenas a *um* quando for iniciada a transferência do segundo canal. Coincidente com este instante resulta também a transição descendente de *LRCK* que sinaliza a transmissão do canal direito, ou canal 2. O sinal *FSYNK* encontra-se de novo no valor lógico *um* durante os vinte e quatro períodos do BCK em que os 24 bits do segundo canal são transmitidos. Após o envio existe uma espera de oito ciclos de *BCK* até começar a próxima transferência. Esta pausa tem a mesma duração no fim da transmissão de ambos canais. Os sinais de dados são colocados na linha *DOUT*. Este ciclo é repetido constantemente se nenhuma interrupção por hardware for feita. Assim sendo, a amostragem e o respectivo envio é periódico e com uma frequência de 96KHz. A Tabela 7 contém o valor do período, frequência e *duty-cycle* dos três sinais de controlo apresentados na Figura 29.

Com o auxílio de um pino do PCM1802 denominado por *Power Down (*PWDN), a FPGA quando se encontra inicializada procede à activação de todos os ADCs em simultâneo. Assim, sabendo que o tempo de *startup* é comum a todos os PCM1802, a amostragem vai ser

feita em simultâneo, uma vez que todas partilham o mesmo relógio. O desfasamento entre os vários sinais vai depender apenas do atraso de propagação.

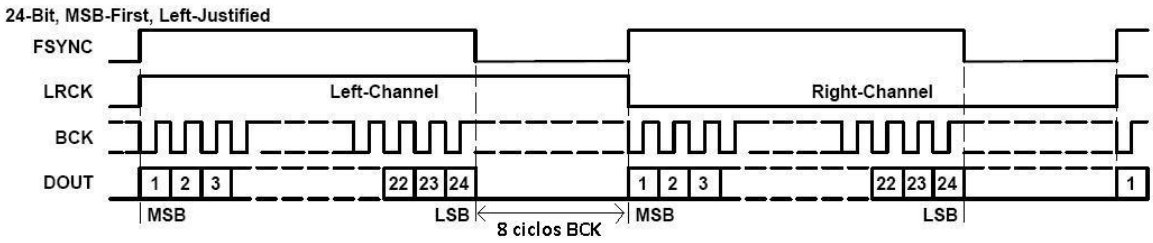


Figura 29 - Diagrama Temporal: PCM1802 [3]

Sinal	Período (ns)	Frequência (kHz)	Duty-Cycle (%)
BCK	$\frac{1}{64 \times \text{fa}} = 162{,}8$	6144	50
LRCK	$\frac{1}{\text{fa}} = 10416{,}7$	96	50
FSYNK	$\frac{1}{2 \times \text{fa}} = 5208{,}3$	192	75
Frequência de Amostragem (f_a) = 96KHz			

Tabela 7 - Sinais de Controlo do PCM1802

4.3.2 – Conversor Digital-Analógico – DAC8534

Descrição do Componente

A conversão de digital para analógico das saídas provenientes da FPGA é feita com recurso a oito conversores DAC8534 [5]. Estes integrados são igualmente da *Burr-Brown Products* e cada um dos conversores tem quatro DACs internos, logo quatro canais de saída como é observável na Figura 30. Assim sendo, a informação entra por uma linha série, *DIN*, passa por um *serial-to-parallel shift register* e é convertida em quatro *resistor-string* DACs diferentes. Este conversor tem uma resolução de 16 bits e suporta uma frequência de relógio máxima de 30MHz. O relógio que controla estes conversores, *SCLK*, é criado na FPGA uma

vez que é esta que controla o envio série da informação. Usando o oscilador embutido na NEXYS2 e através de um divisor de frequência obtém-se um sinal de 25MHz que é usado para esta finalidade.

Embora a resolução do conversor seja de 16 bits, o sinal de entrada tem de ter 24 bits uma vez que são necessários oito bits de controlo para endereçar o integrado, identificar a que canal se refere a informação a ser enviada e se é desejado actualizar a saída imediatamente após a conversão ou esperar pelo último canal para o fazer. Esta trama pode ser observada na Figura 31 onde os primeiros 8 bits são de controlo (*A1, A0, LD1, LD0 DAC Select 1, DAC Select 0 e PD0*) e os restantes 16 de dados (D15 a D0). Todos os bits de controlo têm de ser gerados na FPGA para garantir o funcionamento correcto dos conversores. Como são necessários trinta e dois canais de saída, oito DAC8534 são suficientes.

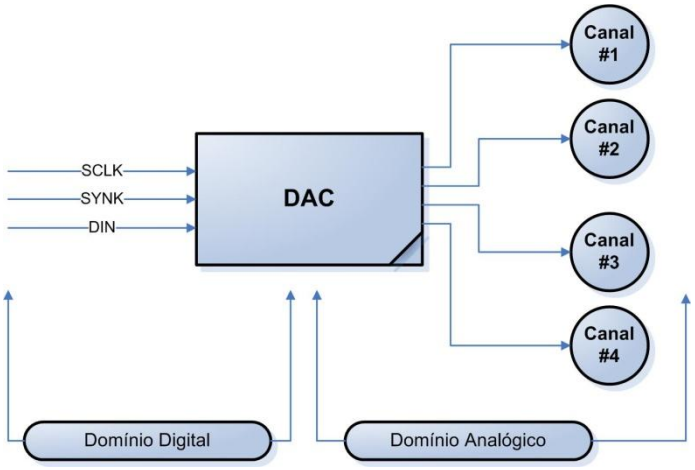


Figura 30 - Conversor Digital-Analógico

DB23											DB12
A1	A0	LD1	LD0	X	DAC Select 1	DAC Select 0	PD0	D15	D14	D13	D12
DB11											**DB0**
D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

Figura 31 - Trama de Envio para DAC [5]

Sincronização FPGA/DAC

Dada a existência de 8 bits adicionais de controlo, para a sincronização entre a FPGA e o conversor analógico-digital apenas é necessário o sinal *SYNK* e o *SCLK*. Neste caso, à semelhança do ADC, o sinal *SYNK* serve para sinalizar o início da transmissão enquanto o sinal *SCLK* serve de relógio para controlar a taxa de transmissão dos bits. O conversor funciona, como é habitual nos conversores digital-analógico, em modo *Slave*. A Figura 32 descreve o comportamento da transferência dos dados via série. O sinal *SYNK* é colocado no valor lógico *um* e, quando ocorrer a transição descendente deste sinal, o *shift register* será activo e os dados deverão ser enviados nos 24 ciclos de relógio seguintes. Cada bit será introduzido no *shift register* do ADC8534 na transição descendente do ciclo de *SCLK*. No 24º ciclo do *SCLK* os bits são bloqueados e usados para controlo, os oito mais significativos, e para dados, os restantes. Esta informação é descodificada sem esperar pela transição ascendente de *SYNK*. A próxima transferência começa na próxima transição descendente de *SYNK*. Este sinal não pode ser trazido a *zero* antes de serem transferidos os 24 bits pois isso provocará uma interrupção e consequente perda de informação. Entre o 24º bit e a próxima transição descendente de *SYNK* tem obrigatoriamente de ocorrer uma espera mínima de t_9=130ns. Quatro ciclos de espera do sinal de 25MHz que corresponde a 160ns são suficientes para respeitar este requisito.

Sendo necessários oito DAC8534, todos eles partilharão a linha de relógio e a do sinal *SYNK*. A única linha que é independente para cada conversor é a *DIN*, uma vez que cada um terá informação distinta.

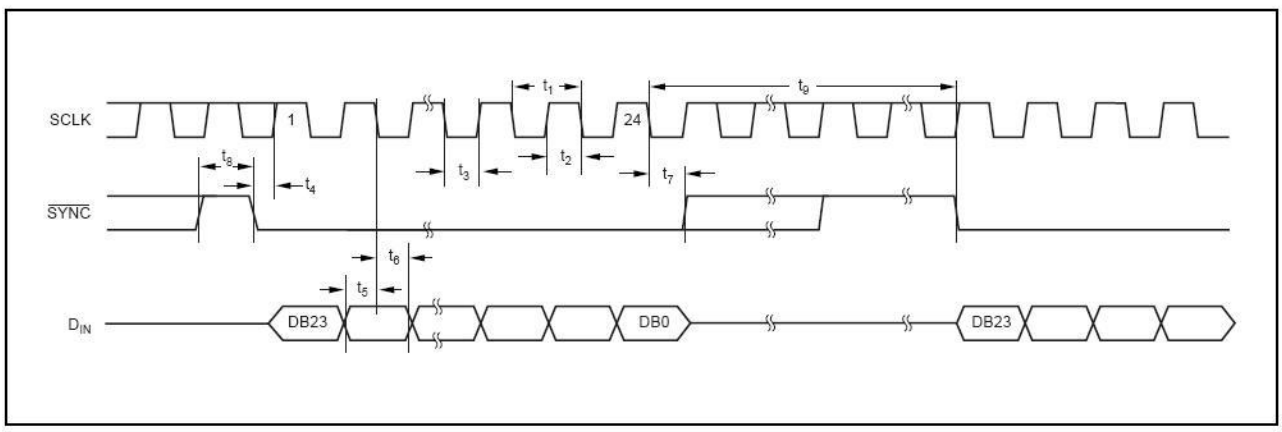


Figura 32 - Diagrama Temporal: DAC8534 [5]

4.3 – Fonte de Alimentação

A fonte de alimentação foi dimensionada de forma a respeitar os requisitos do sistema referentes à tensão e corrente pretendida. A fonte alimenta directamente algum hardware com recurso a dois reguladores que permitem a criação de +12V e -12V. Usando um regulador adicional alimenta-se a placa NEXYS2 com 5V, sendo que esta gera uma tensão de 3.3V que é necessária para o seu funcionamento assim como para alimentar os conversores analógico-digital e também o oscilador usados para estes conversores. Assim sendo, não foi necessário criar esta voltagem usando um quarto regulador.

Os buffers colocados à entrada e saída do sistema para adaptar o sinal ao exterior (secção 4.4.1 – Buffers de Entrada e de Saída – INA137 e DRV134) necessitam de +/-12V logo adquiriu-se um transformador de 220V/15V@2A que foi rectificado para criar os +/-12V. Para regular a alimentação positiva escolheu-se o LM7812 [14] enquanto que para a negativa se seleccionou o MC7912 [15]. O LM7805 [11] foi o componente escolhido para gerar os 5V que alimentam a placa NEXYS2, os amplificadores operacionais usados para filtrar o sinal e todos os conversores presentes no sistema. Montou-se um rectificador de onda completa (ponte de díodos) para garantir a rectificação da tensão proveniente do transformador.

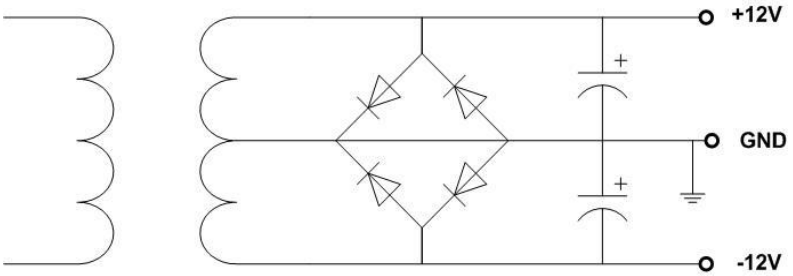


Figura 33 - Rectificação

Para além da corrente que a fonte tem de fornecer, outra preocupação que se teve relaciona-se com a dissipação de calor. Uma vez que o sistema total pode atingir temperaturas elevadas e o ambiente em que se insere pode não providenciar muita refrigeração calculou-se e escolheu-se convenientemente um dissipador para que o material não tenha danos desta natureza. Para tal é igualmente necessário determinar o consumo do sistema.

A Tabela 8 representa o consumo máximo previsto para os vários componentes assim como o consumo total estimado do sistema. Com o recurso à documentação de cada um dos componentes, estimou-se a corrente máxima de cada um. Dando alguma margem de erro e sabendo o número de cada componente existente no sistema, estimou-se o consumo total.

CONSUMO MÁXIMO ESTIMADO DO SISTEMA			
Componente	Consumo máximo estimado por unidade (mA)	Quantidade	Total (mA)
MCP	0,33	40	13,00
INA137	5,80	4	23,20
DRV134	25,00	32	800,00
ADC	51,71	4	206,84
DAC	2,23	8	17,80
FPGA	350,00	1	350,00
		TOTAL	1410,84

Tabela 8 - Consumo Máximo Estimado do Sistema

Dada estimação do consumo de pico do sistema, a fonte foi dimensionada para fornecer 2A enquanto que o projecto de dissipação foi feito considerando que circulava uma corrente de 1.5A. Todo o consumo máximo é calculado com margem de erro, logo a fonte utilizada é suficiente.

Apresenta-se agora o projecto de dissipação que permite escolher um dissipador adequado ao sistema. A equação seguinte representa o cálculo da tensão de *Ripple* no transformador. Este valor representa a variação da tensão de alimentação na saída do rectificador:

$$V_{ripple} = \frac{I_{max} \times dt}{C} = \frac{2 \times {}^{1}/_{(2 \times 50)}}{4{,}7 \times 10^{-3}} = 4{,}255V$$

Sabendo a tensão de pico do transformador, a queda no díodo e a tensão de *Ripple*, é possível determinar entre que valores oscilará a tensão de entrada dos reguladores:

$$V_{pico} - V_{díodo} - V_{ripple} < V_{out} < V_{pico} - V_{díodo}$$
$$15\sqrt{2} - 0{,}7 - 4{,}255 < V_{out} < 15\sqrt{2} - 0{,}7$$
$$16{,}26 < V_{out}\,(V) < 20{,}51$$

A potência dissipada no regulador é dada por:

$$P = \Delta_V \times I = 2 \times (20{,}51 - 16{,}26) \times 1{,}5 = 12{,}75 \approx 13W$$

A variação de temperatura do dissipador é dada pela seguinte expressão. Aqui admite-se que a temperatura exterior atinge no máximo 50ºC enquanto que se pretende uma temperatura máxima na caixa de 130ºC.

$$\Delta_T = T - 50℃ = 130 - 50 = 80℃$$

Tendo a variação da temperatura, é possível determinar as resistências necessárias para garantir uma boa dissipação. R_{JC} é a resistência entre a junção e a caixa e é fornecida no datasheet do regulador LM7812 [14]. R_{CD} representa a resistência entre a caixa do regulador e o dissipador e pode ser desprezada. R_{DA} é a resistência entre o dissipador e o ar e é o parâmetro que influencia a escolha do dissipador.

$$\Delta_T = P \times \left(R_{JC} + R_{CD} + R_{DA}\right)$$
$$80 = 13 \times (4 + R_{DA})$$
$$R_{DA} = 2{,}15\ ℃/W$$

Obtida uma resistência dissipador/ar de 2,15ºC/W foi escolhido um dissipador cuja resistência é de 2,1ºC/W. Para garantir uma margem de segurança maior, foram colocados dois destes dissipadores no regulador de 12V uma vez que é onde circula mais corrente, logo, tem probabilidade de aquecer mais.

4.4 – Outros Componentes

Juntamente com o hardware apresentado até agora foram necessários alguns componentes que tratassem do acondicionamento do sinal. A adaptação destes sinais às entradas e às saídas e a redução de ruído são os tópicos abordados de seguida.

4.4.1 – Buffers de Entrada e de Saída – INA137 e DRV134

Os canais áudio na entrada chegam em formato diferencial uma vez que respeitam a norma XLR [29] que é tipicamente usada nos sistemas áudio para que a linha em que o sinal é transmitido seja balanceada [19] dado que isso permite o uso de cabos de longo comprimento reduzindo a susceptibilidade a interferências externas. É conveniente que a saída seja igualmente disponibilizada nesse mesmo formato para que a integração em ambientes áudio seja proporcionada.

Para este efeito são utilizados buffers à entrada que convertem o sinal de diferencial para unipolar enquanto que na saída os buffers escolhidos fazem o inverso, i.e., convertem o sinal de unipolar para diferencial.

O buffer INA137 é um receptor áudio diferencial e o DRV134 é um driver de áudio balanceado. Uma vez que estes integrados foram desenvolvidos especificamente para aplicações áudio, são a escolha ideal para este projecto.

A configuração do buffer de entrada é a observada na Figura 34. Esta configuração tem um ganho de ½. Por sua vez, o buffer de saída foi configurado com um ganho de 2, como é possível observar na Figura 35. Informação adicional sobre os componentes pode ser encontrada na respectiva documentação [4] [2].

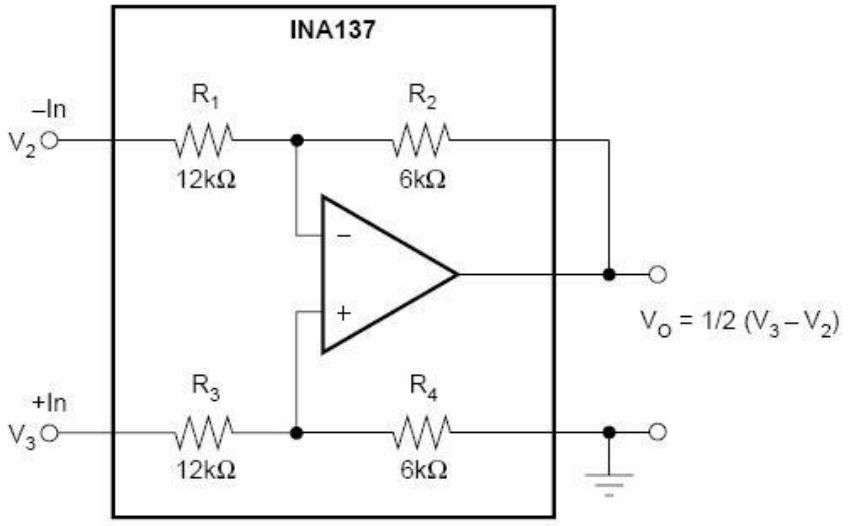


Figura 34 - Buffer de Entrada - INA137 [4]

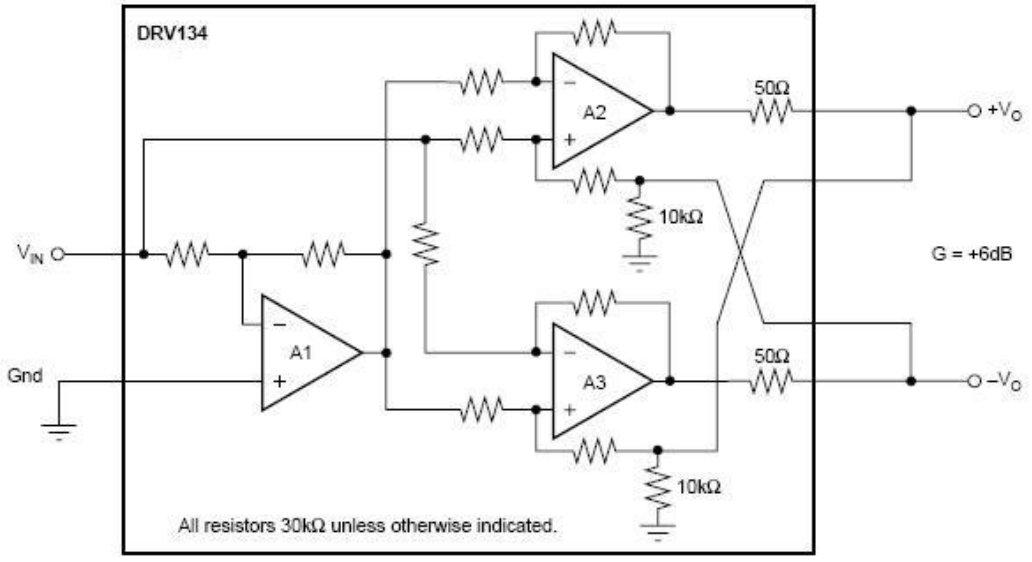


Figura 35 - Buffer de Saída - DRV134 [2]

4.4.2 – Filtros Passa-Baixo

Para eliminar ruído que possa aparecer fora da gama áudio foram introduzidos, em cada um dos canais de entrada e de saída, filtros passa-baixo. Dimensionou-se então um filtro passa-baixo de segunda ordem com uma topologia de *Sallen-Key* representado na Figura 36 cuja frequência de corte (fc) é dada por (1).

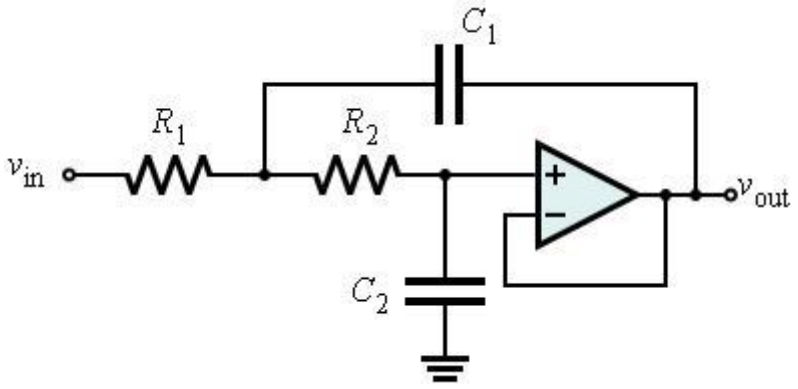


Figura 36 - Filtro Passa-Baixo com topologia Sallen-Key

$$fc = \frac{1}{2\pi \times \sqrt{R_1 \times R_2 \times C_1 \times C_2}} \qquad (1)$$

Como amplificadores operacionais (OPAMPs) foram usados vários MCP602s na entrada e MCP604s na saída dado que os ADCs têm dois canais e os DACs quatro, logo, para cada conversor é usado um integrado de amplificadores operacionais. Relembra-se que o componente MCP602 tem dois amplificadores operacionais embutidos enquanto que o MCP604 tem quatro.

4.5 – Interligação do Hardware do Sistema

A Figura 37 representa um andar completo do sistema. Verifica-se mais uma vez que o sinal de entrada (E) é ligado ao buffer INA137, é de seguida filtrado com recurso a um MCP602 e convertido para digital por um PCM1802 que envia a informação para a FPGA presente na placa NEXYS2. Uma vez geradas as saídas na FPGA, os sinais são convertidos para analógico usando os DAC8534s, filtrados com auxílio a MCP604s e colocados na saída usando os DRV134s.

As ligações entre a FPGA e os conversores foram feitas aproveitando o conector Hirose e dois PMODS de doze pinos. O conector Hirose tem pinos suficientes para todas as ligações necessárias entre os conversores e a NEXYS2 mas foi necessário recorrer aos PMODS para garantir os 5V que alimentam os DACs e a parte analógica dos ADCs. Enquanto o Hirose apenas fornece 3.3V, os PMODS possibilitam ambas as tensões. Para tal basta trocar um *jumper* presente na NEXYS2 que define se aquele módulo terá lógica entre 0 e a tensão de alimentação da placa ou entre 0 e 3.3V.

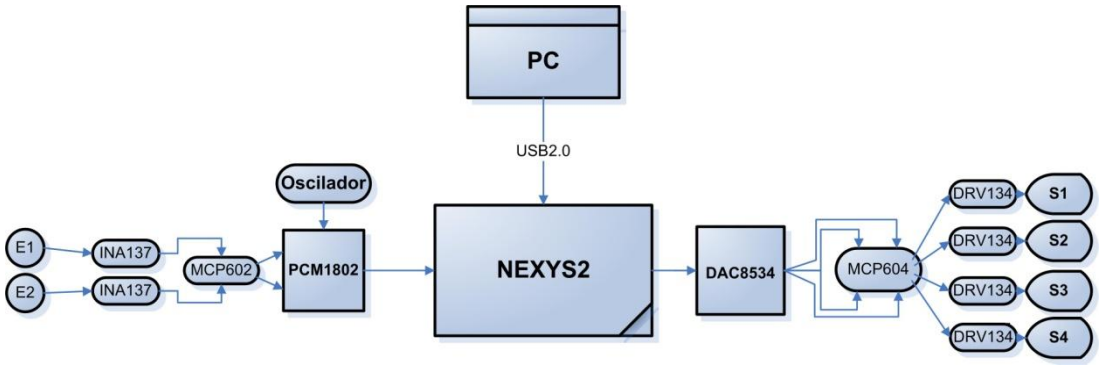


Figura 37 - Andar Completo do Sistema

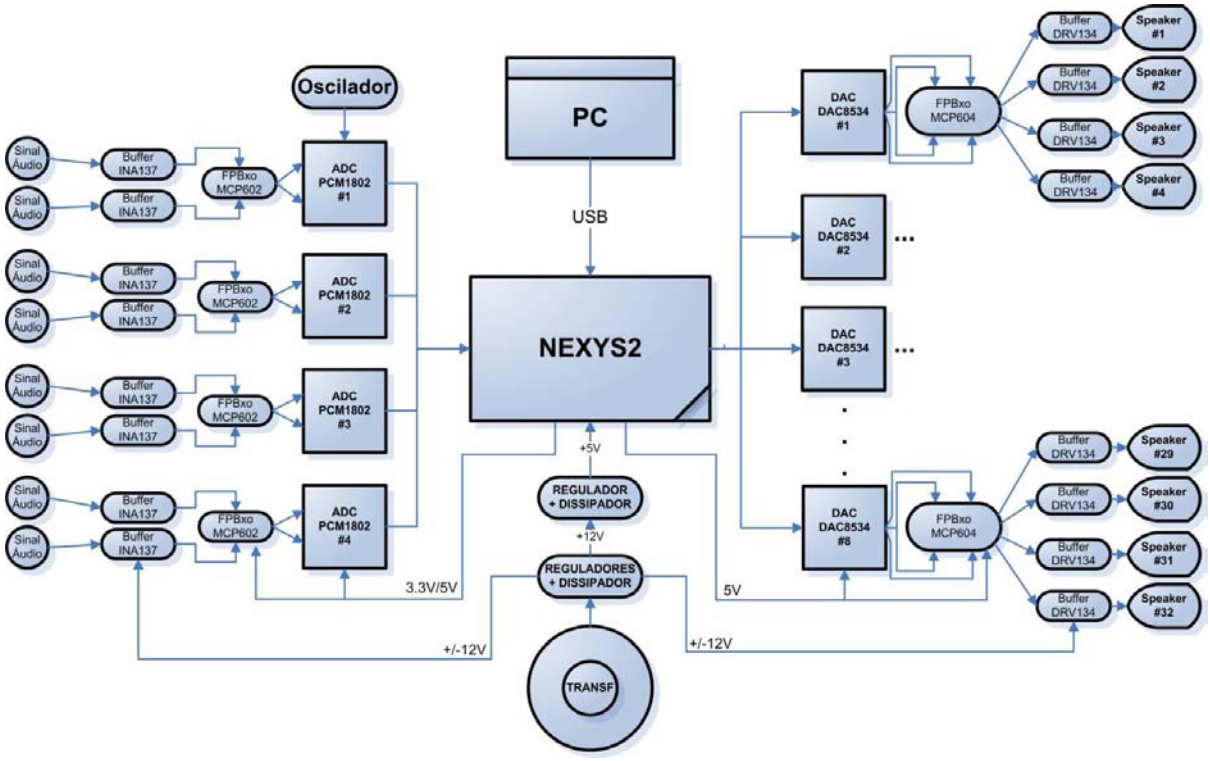


Figura 38 - Esquema completo do Sistema

O andar observado é replicado até se obterem o número de entradas e saídas desejados. A Figura 38 representa o esquema completo assim como a distribuição da alimentação pelo circuito.

4.6 – Placa de Circuito Impresso

4.6.1 – Placa de Testes

Para testar a configuração dos conversores analógico-digital e digital-analógico assim como a sua interacção com a FPGA, foi dimensionado um *Printed Circuit Board (*PCB) que englobasse um oscilador de 24,576MHz, um ADC, um DAC e as respectivas ligações com a FPGA através dos conectores PMOD. A Figura 39 representa o esquema deste sistema de testes. Os filtros foram dimensionados posteriormente e testados em placa branca estando ligados ao PCB. Desta forma, conseguiu-se observar o comportamento do integrado PCM1802 e verificar a sua configuração assim como a comunicação implementada em modo *Master*. Tendo este sistema inicial dois canais de entrada e quatro de saída implementaram-se diferentes algoritmos na FPGA de forma a que se permitissem fazer várias experiências com estes canais. Nesta fase implementou-se também a comunicação USB que permitiu de seguida definir, através do computador, o peso de cada uma das duas entradas em cada saída. A placa NEXYS2 era nesta fase alimentada através da ligação USB. Assim sendo, nos conectores PMOD existia a possibilidade de alimentar o hardware quer com 5V quer com 3.3V. Foi usada esta configuração uma vez que nesta fase a corrente necessária era inferior à máxima possível que a unidade USB pode providenciar (500mA).

A placa de circuito impresso que contém o oscilador e os conversores foi desenhada usando a ferramenta Eagle® e o respectivo esquemático e *footprint* pode ser encontrado em anexo (Anexo E – Placas de Circuito Impresso).

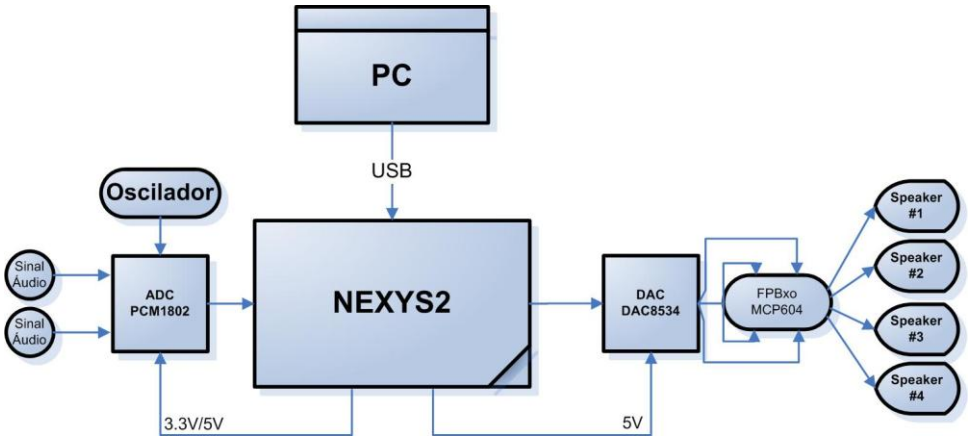


Figura 39 - Esquema do Circuito de Testes

4.6.2 – Placa Final

A placa final foi desenvolvida da mesma forma que o circuito de testes mas desta vez a englobar todos os componentes utilizados no projecto. Nesta fase foram já introduzidos todos os componentes contabilizando no total quatro *buffers* de entrada INA137, quatro integrados de amplificadores operacionais MCP602 e oito MCP604, quatro conversores analógico-digital PCM1802, oito conversores digital-analógico DAC8534, trinta e dois *buffers* de saída DRV134 e dois reguladores, um LM7812 e outro MC7912. Como se observa pela Figura 40, o PCB desenvolvido inclui ligações à placa NEXYS2, ao transformador e aos canais de entrada e saída. Dada a dimensão do sistema optou-se por desenhar todo este circuito em duas placas de circuito impresso separadas. Cada uma contém quatro canais de entrada e dezasseis canais de saída. O *footprint* de cada uma das placas pode ser encontrado em anexo (Anexo E – Placas de Circuito Impresso).

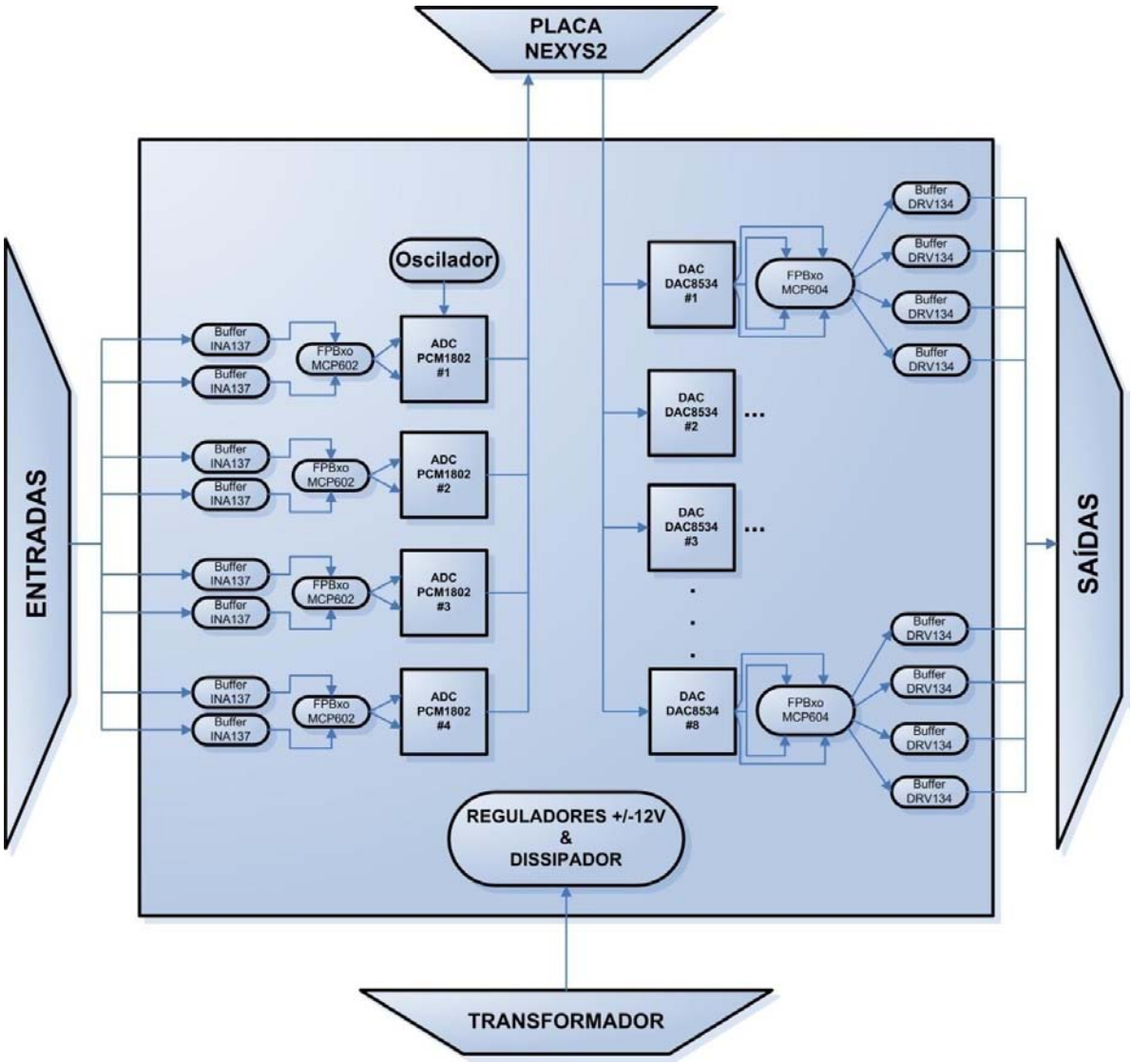


Figura 40 - Esquema da Placa de Circuito Impresso Final

4.7 – Montagem Final

O sistema final encontra-se representado na Figura 41. Foi desenvolvida uma placa adicional que permite converter algumas das ligações PMOD provenientes do PCB principal no formato do conector *Hirose*. Este circuito inclui também a regulação de 5V que alimenta a placa NEXYS2.

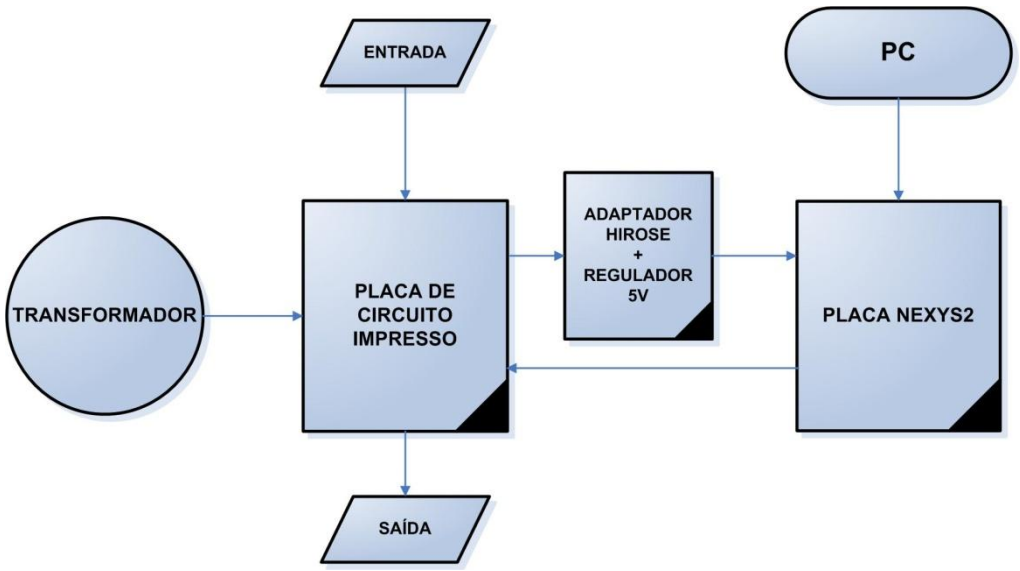


Figura 41 - Sistema Final

A Figura 42 representa uma fotografia do sistema final. Aqui é possível observar a placa de circuito impresso (PCB) (à esquerda), o PCB auxiliar (devidamente assinalado) e a placa NEXYS2 que contém a FPGA. Nesta imagem é possível observar, no PCB principal os componentes responsáveis pela rectificação e regulação da alimentação, as entradas e as saídas. São também visíveis os componentes responsáveis pelos filtros passa-baixo colocados nas saídas assim como os buffers de saída. Junto às entradas encontram-se os buffers de entrada seguidos dos filtros passa-baixo. Na camada inferior da placa de circuito impresso (não visível na figura) estão soldados os conversores analógico-digital e digital-analógico. Existe uma segunda placa semelhante à observada que, estando colocada por baixo, contém os restantes canais. A segunda placa diferencia-se apenas na medida em que não tem electrónica para a alimentação.

O PCB auxiliar contém, como já foi referido, a conversão de conectores PMOD para *Hirose* assim como a regulação de +12V para +5V. Os cabos visíveis na figura são usados para

transferir informação da FPGA para a placa de circuito impresso assim como no sentido inverso. Existem ligações tanto no PCB auxiliar como directamente nos PMOD assinalados.

Por último, é possível identificar, na placa NEXYS2, a ligação USB que permite o envio dos coeficientes através de um computador.

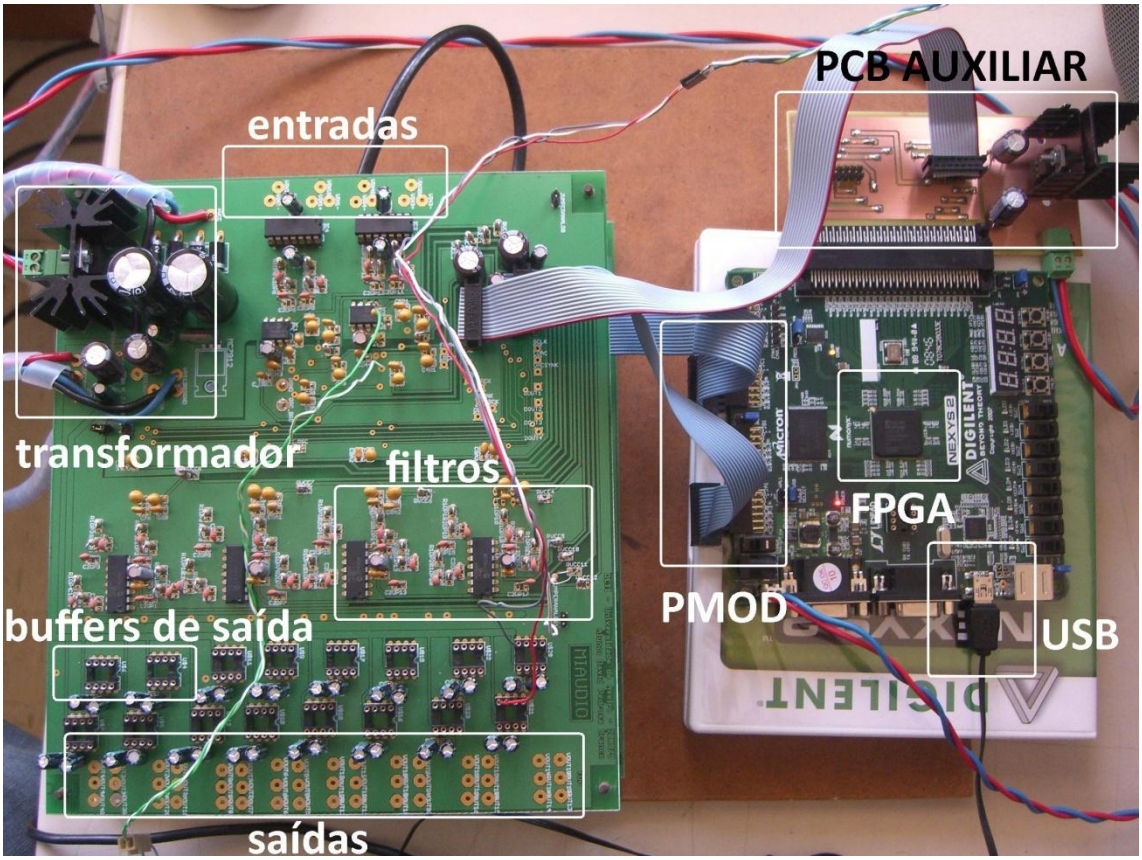


Figura 42 - Sistema Final – Fotografia

Capítulo 5 – TESTES E RESULTADOS

5.1 – Diagrama Temporal

A Figura 43 ilustra a aquisição de alguns sinais feita no arranque do sistema. Nesta fase de testes a frequência de relógio usada para controlar a lógica da FPGA foi de 25MHz. Este diagrama foi criado usando um analisador lógico digital e avaliando vários sinais presentes na placa do sistema assim como sinais internos da FPGA que foram disponibilizados em pinos de saída da placa NEXYS2 para efeitos de depuração. A Tabela 10 apresenta uma breve descrição dos sinais adquiridos.

Neste diagrama é possível identificar alguns intervalos de tempo que caracterizam o funcionamento do sistema. O intervalo de tempo t_1 representa o atraso entre o instante em que o conversor analógico-digital (ADC) enviou a sua primeira amostra e o instante em que a FPGA termina o envio da amostra respectiva de saída para o conversor digital-analógico (DAC), ou seja, t1 representa o tempo de processamento de uma amostra. Assim sendo, o atraso entrada/saída é igual a este tempo mais o tempo de conversão de cada um dos conversores, ADC e DAC.

Observando os intervalos de tempo t_2 e t_3, verifica-se que, tanto o bloco Aritmético como o bloco de Saída, têm uma periodicidade de funcionamento igual ao período de amostragem do ADC. Os diferentes blocos da lógica da FPGA têm também um período de funcionamento igual ao de amostragem do sinal de entrada.

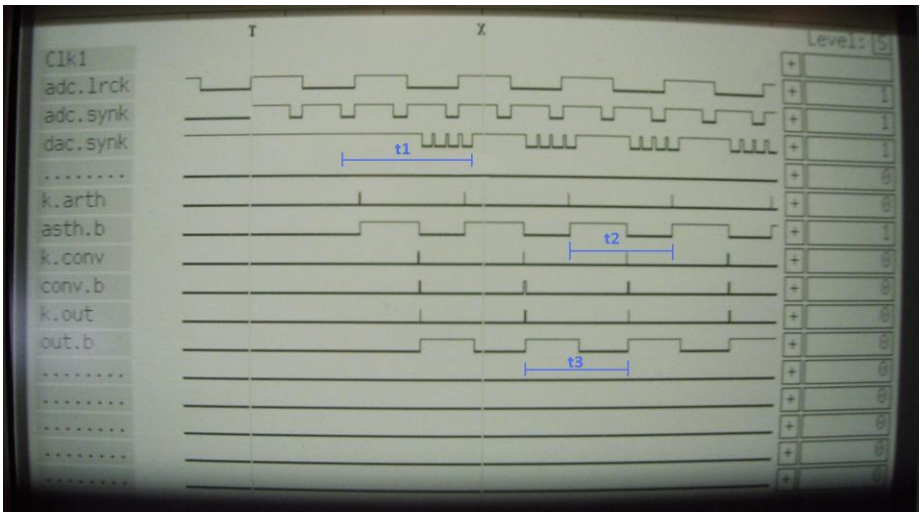


Figura 43 - Diagrama Temporal de Teste

	Intervalo de Tempo (µs)	Descrição
t_1	13.085	Tempo de Processamento
t_2	10.4	Intervalo de Tempo entre activações do Bloco Aritmético
t_3	10.4	Intervalo de Tempo entre activações do Bloco de Saída

Tabela 9 – Intervalos de Tempo do Diagrama Temporal de Teste

Sinal	**Descrição**
adc.lrck	Sinal LRCK proveniente do ADC
adc.synk	Sinal FSYNK proveniente do ADC
dac.synk	Sinal SYNK proveniente do ADC
k.arth	Kick Arithmetic – Sinaliza o arranque do bloco Aritmético
arth.b	Arithmetic Busy – Sinaliza o estado do bloco Aritmético (0 – livre; 1 – ocupado)
k.conv	Kick Converter – Sinaliza o arranque do bloco de Conversão de Formato
conv.b	Conveter Busy – Sinaliza o estado do bloco de Conversão de Formato (0 – livre; 1 – ocupado)
k.out	Kick Output – Sinaliza o arranque do bloco de Saída
out.b	Outputs Busy – Sinaliza o estado do Bloco de Saída (0 – livre; 1 – ocupado)

Tabela 10 - Descrição dos Sinais do Diagrama Temporal de Teste

5.2 – Atraso do Sistema

Para determinar o atraso do sistema da entrada para a saída, foi introduzida uma onda sinusoidal de 1kHz e enviado um coeficiente que permitiu colocar a entrada 1, onde foi aplicada a onda, na saída 1. Analisando a passagem por zero, verificou-se que existe um atraso entrada/saída de aproximadamente 210µs. Este atraso deve-se ao tempo de conversão do sinal analógico para digital, tempo de processamento de cada amostra por parte da FPGA e à conversão da saída de digital para analógico.

O resultado apresentado é satisfatório uma vez que este atraso não é perceptível ao ouvido humano. No entanto a grandeza deste atraso deve-se também ao efeito do filtro passa-baixo uma vez que este provoca um atraso de fase na onda de saída em relação à de entrada. O valor obtido é de uma grandeza que dificilmente é atingido noutro tipo de implementação. O facto de a mistura ser feita em hardware na FPGA permite garantir que o atraso entrada/saída é constante e extremamente reduzido.

A Figura 44 e Figura 45 representam a leitura do atraso do sistema feita num osciloscópio digital.

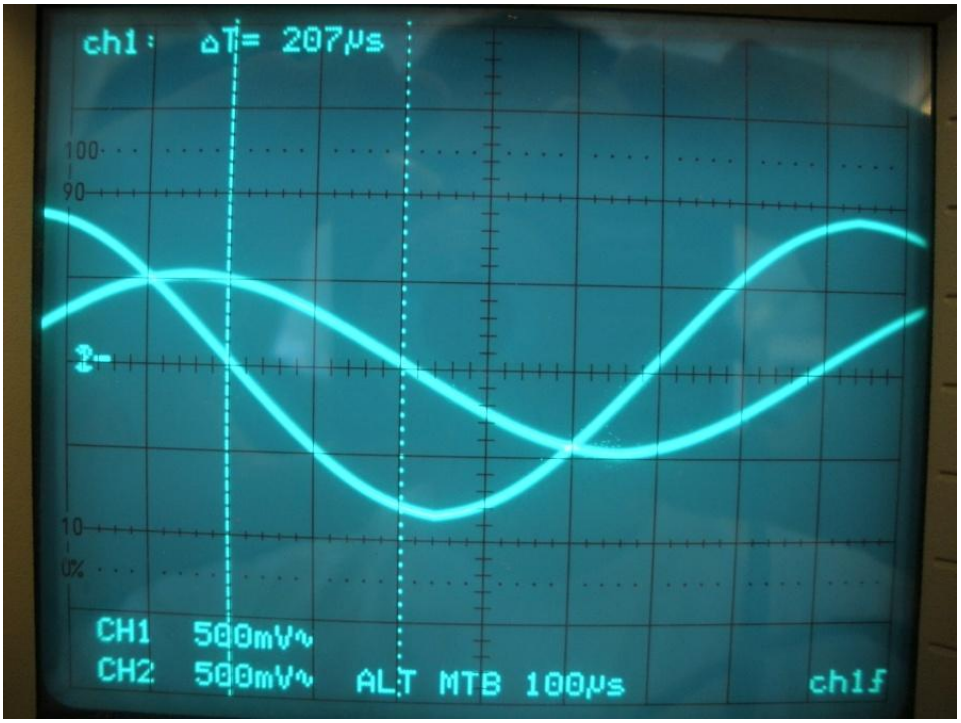


Figura 44 - Atraso do Sistema

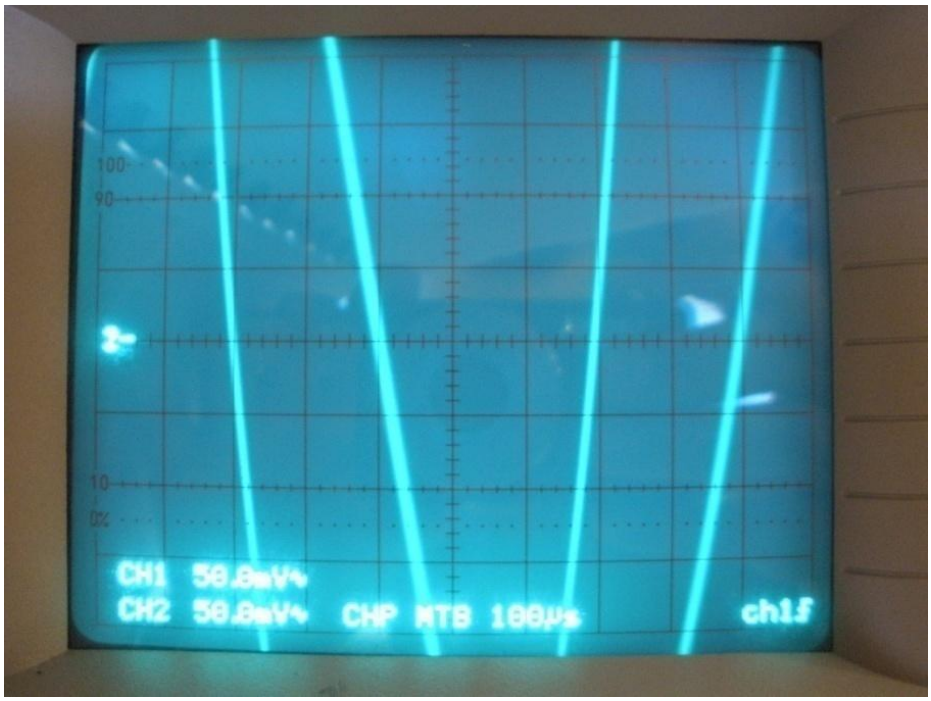


Figura 45 - Atraso do Sistema (Ampliação)

5.3 – Ganho e Frequência de Corte do Sistema

Uma vez que o ganho do sistema é dependente do coeficiente enviado pelo computador foram feitos testes para determinar qual o coeficiente que representa um ganho unitário e qual o máximo ganho possível.

Os pontos onde foram feitas as medições são ilustrados na Figura 46. Para garantir que o filtro não influência a amplitude do sinal, a sinusóide aplicada encontra-se na gama de passagem. A entrada (E) foi colocada directamente na entrada do conversor analógico-digital (ADC) e a saída (S) foi lida após o filtro passa-baixo.

Relembrando que os coeficientes enviados são de 16 bits e que estes se encontram em complemento para dois, ao enviar o coeficiente com o valor 0x7FFF, foi obtido um ganho de 1.753. Ajustando o coeficiente até se encontrar um ganho unitário chegou-se ao valor do coeficiente 0x4900. Para o valor de ganho do sistema completo há ainda que considerar que os buffers de entrada têm um ganho de ½ enquanto que os de saída têm um ganho de 2.

Figura 46 - Ganho do Sistema

Entrada (V)	Saída (V)	Coeficiente (hexadecimal)	Ganho
1.78	3.12	0x7FFF	1.753
1.78	1.78	0x4900	1

Tabela 11 - Ganho do Sistema

Para a determinação da frequência de corte foi aplicada uma sinusóide à entrada cuja frequência se encontra dentro da gama de passagem dimensionada e, aumentando a frequência de entrada, observou-se quando é que a saída atingia uma atenuação de -3dB. A frequência de corte obtida foi de aproximadamente 20kHz.

5.4 – Análise Espectral

A Figura 47 representa a análise espectral numa escala logarítmica do sinal de entrada (subgráfico superior) assim como o de saída (subgráfico inferior). Foi gerada uma sinusóide de 1kHz em MATLAB®, reproduzida e gravada numa entrada da placa de som do computador. O ficheiro wav gerado na gravação foi analisado no MATLAB® de forma a gerar os gráficos aqui apresentados. Foi usado este método para que se tivesse em consideração o ruído introduzido pela gravação. Considerando isto, foi aplicado o mesmo sinal na entrada do sistema. Configurando a matriz para dar um ganho unitário, foi gravada a saída cuja análise espectral se encontra no subgráfico inferior da Figura 47. Como se pode observar, a análise espectral dos sinais de entrada e saída é semelhante. No sinal adquirido na saída verifica-se o aumento de algum ruído mas no entanto encontram-se ambos perto dos -90dB.

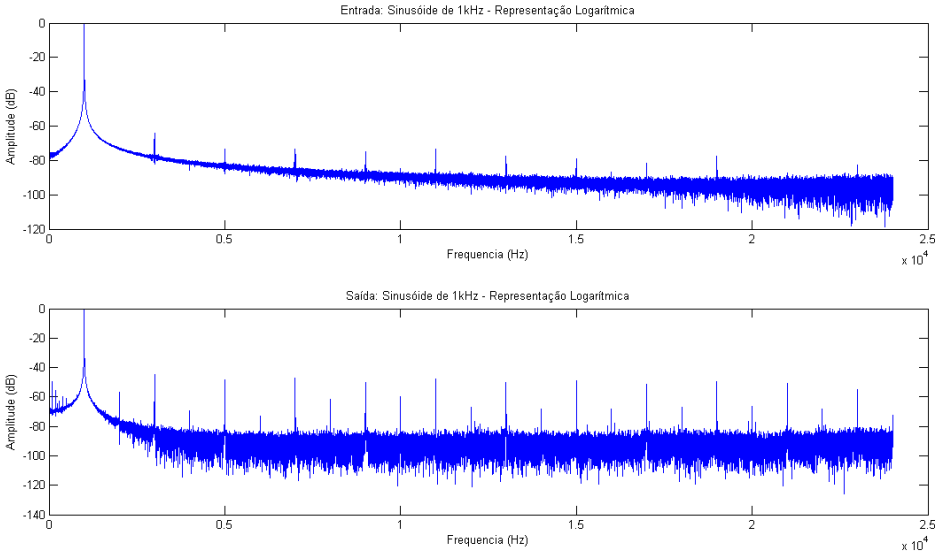


Figura 47 - Análise Espectral do Sinal de Entrada e Saída

O sinal de saída foi também analisado com recurso ao equipamento *Precision One*. Aqui foi de novo confirmada uma frequência de corte de, sensivelmente 20KHz e foi medida a distorção harmónica mais o ruído (THD + N). O valor obtido foi de 0,09% o que representa uma medição bastante satisfatória. Esta medição foi feita com um sinal de entrada com amplitude de 1,28V@1KHz.

5.5 – Consumo do Sistema

O consumo do sistema foi medido colocando um amperímetro em série com a alimentação. Foram analisados três cenários como demonstrado na Figura 48.

Na primeira medição, foi colocado o amperímetro em série com a alimentação da placa NEXYS2 que por sua vez alimentava os doze conversores (quatro ADCs e oito DACs) e os amplificadores operacionais que formam os filtros passa-baixo (quatro MCP602 e oito MCP604). Mediu-se uma corrente que rondava os 395mA. Aplicando um sinal à entrada e colocando-o em todas as saídas, a corrente aumentou cerca de 2mA.

O cenário seguinte consistiu em não ligar a NEXYS2, ficando apenas todos os buffers de entrada e saída alimentados. Nesta medição obteve-se uma corrente de cerca de 296mA. Nesta situação, assim como na seguinte, o amperímetro foi colocado em série com o transformador sendo medida a corrente alterna.

Por fim todo o sistema foi alimentado e colocado um sinal em todas as saídas. A corrente resultante foi de 600mA.

Verifica-se assim que, uma vez que o transformador é de 2A, a corrente do sistema será sempre inferior à máxima possível.

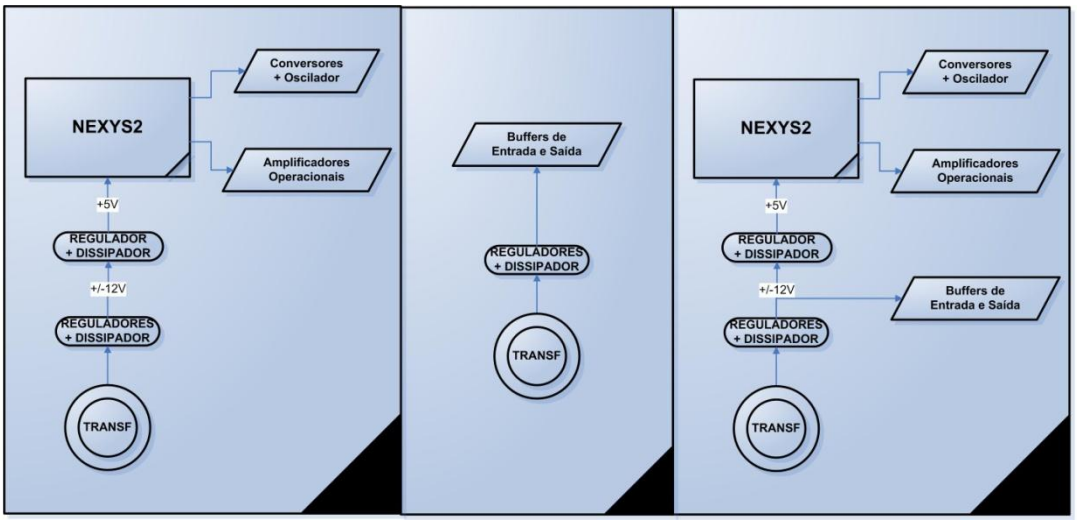


Figura 48- Consumo do Sistema: Cenário I, II e III

5.6 – Recursos da FPGA

Avaliando o projecto na sua totalidade, verifica-se que a FPGA usada tem bastantes recursos livres sendo assim possível o uso de mais lógica para introdução de novas funcionalidades.

São usados apenas quatro dos vinte bancos de memória dedicados disponíveis. Os dezasseis multiplicadores usados podem ser reduzidos em metade, i.e., é necessária apenas a utilização de oito multiplicadores. Para tal, a máquina de estados finita implementada no bloco aritmético deverá ser alterada para gerar apenas uma amostra de saída e não duas em cada iteração. Isto é apenas é garantido se for usado o oscilador de 50MHz como relógio da FPGA e não uma divisão do mesmo. Isto deverá ser feito para que o tempo de processamento de uma amostra continue inferior à taxa de amostragem dos conversores analógico-digital. O valor de 13,085μs apresentado anteriormente permanecerá sensivelmente o mesmo uma vez que a mudança para 50MHz implica uma redução para metade enquanto que o uso de apenas oito

multiplicadores duplica o tempo de funcionamento do bloco aritmético que representa o bloco mais demorado.

No que respeita a Look-up Tables (LUT's) verifica-se que são usadas apenas cerca de 26% das existentes, enquanto que a nível de número de Flip Flops são usados 49% dos disponíveis. Este valor é o estimado pela ferramenta ISE versão 11.3 da Xilinx no qual foi gerado o ficheiro de configuração da FPGA. Relembra-se que esta FPGA não é das mais potentes no mercado nem mesmo da sua família, uma vez que existe uma versão que contém mais do dobro dos Configurable Logic Blocks (CLB's). Está também em aberto a possibilidade de optimizar a lógica usada em alguns pontos do sistema como, por exemplo, a inclusão da operação de conversão de formato no bloco Aritmético. Em anexo existe uma tabela que apresenta alguns detalhes sobre a taxa de utilização da lógica da FPGA.

Capítulo 6 – CONCLUSÕES E TRABALHO FUTURO

6.1 – Conclusões

O sistema desenvolvido encontra-se actualmente em pleno funcionamento. Foram conseguidos 8 canais de entrada e 32 de saída e foi implementada a comunicação com o PC que permite o envio dos parâmetros para a realização da mistura de áudio. A nível de sinal foi conseguida uma qualidade que viabiliza a utilização de MIAUDIO.

A escolha dos conversores analógico-digital e digital-analógico foi trabalhosa e importante para o resto do projecto. Escolheram-se conversores que se adaptam bem ao projecto e foi possível estabelecer a sua comunicação com a FPGA. Com o restante hardware foi conseguida uma filtragem do sinal e uma adaptação ao exterior de forma a possibilitar uma integração em sistemas áudio.

O atraso entrada/saída obtido é um resultado bastante importante dado que apenas neste tipo de implementação (mistura realizada em hardware em tempo-real) é garantido um intervalo de tempo perfeitamente constante e extremamente reduzido. Este é um aspecto diferenciador em MIAUDIO.

As expectativas iniciais quanto ao custo do sistema e tempo de desenvolvimento foram atingidas. O tempo de desenvolvimento foi relativamente curto, uma vez que o projecto foi desenvolvido por uma pessoa em sensivelmente 9 meses. O custo do projecto, valor que ronda os 500€ (Anexo C – Custo do Sistema), foi reduzido quando comparado com sistemas semelhantes. A diferenciação deste sistema para os que adoptaram uma solução por software reside no facto do sistema operativo (SO) que envia os parâmetros da mistura precisa de muito poucos recursos para o fazer. O SO fica disponível para realizar outras tarefas. MIAUDIO é flexível a alterações, a introdução de novas funcionalidades e o seu algoritmo é reconfigurável. É possível a introdução de novas funcionalidades sem que sejam necessárias alterações no hardware desenvolvido.

O *know-how* aqui envolvido pode ser facilmente adaptado para a criação de mesas de mistura digitais. É também possível acrescentar outros tipos de processamento no sinal como *delays* e outros tipos de efeitos dado que a FPGA tem um grande poder de processamento.

6.2 – Trabalho Futuro

A continuação do desenvolvimento de MIAUDIO assenta principalmente na integração da comunicação USB noutras ferramentas de software. Actualmente existe um projecto criado em Visual C++ que permite o envio dos coeficientes da matriz em ambiente Windows® através de uma linha de comandos DOS. Com as ferramentas actuais é possível a criação de uma *Dynamic Link Library* (DLL) e utilização da mesma em software de alto nível. Isto permitirá a ferramentas como o *Pro Tools*, *SuperCollider* e MAX/MSP controlarem a mistura feita na FPGA. Para tal será conveniente a adaptação da comunicação para MAC OS.

A introdução de novas funcionalidades neste sistema é também uma perspectiva futura. A forma como os coeficientes são caracterizados pode ser diversificada. Podem ser usados, por exemplo, sensores de ultra-sons cuja posição é lida e, consoante a mesma, é definido o peso de canais de entrada nas saídas desejadas. A introdução de periféricos que interajam quer com a FPGA quer com o computador pode ser uma forma de caracterizar os parâmetros da matriz de mistura de áudio.

Este projecto é flexível e receptível a novas funcionalidades. A imaginação permitirá o surgimento de novas ideias que poderão certamente ser implementadas.

BIBLIOGRAFIA

[1] Ashenden, P. J. (2008). The Designer's Guide to VHDL, Third Edition, Morgan Kaufmann.

[2] Burr-Brown (April 2007). DRV134. Audio Balanced Line Drivers.

[3] Burr-Brown (January 2005). PCM1802. Single-Ended Analog-Input 24-BIT, 96-kHz Stereo A/D Converter.

[4] Burr-Brown (July 2007). INA137. Audio Differential Line Receivers.

[5] Burr-Brown (September 2002). DAC8534. Quad Channel, Low Power, 16-Bit, Serial Input, Digital-to-Analog Converters.

[6] Digilent (June 2008). Digilent Nexys2 Board Reference Manual.

[7] Digilent (March 2005). Digilent Port Communications Programmers Reference Manual.

[8] Digilent (October 2004). Digilent Parallel Interface Model Reference Manual.

[9] Digilent (September 2004). Digilent USB 2 Module Reference Manual.

[10] EuroQuartz X091 Series Oscillators.

[11] FairChild (2001). MC78XX/LM78XX/MC78XXA. 3-Terminal 1A Positive Voltage Regulator.

[12] McCartney, J. SuperCollider: A New Real Time Synthesis Language. http://www.audiosynth.com/icmc96paper.html.

[13] Microchip (2007). MCP601/1R/2/3/4. 2.7V to 6.0V Single Supply CMOS Op Amps.

[14] NationalSemiconductor (May 2000). LM78XX. Series Voltage Regulators.

[15] OnSemiconductor (August 2006). MC7900 Series. Negative Voltage Regulators.

[16] Ott, H. W. (1988). Noise Reduction Techniques in Electronic Systems, Second Edition, NoiseJohn Wiley & Sons.

[17] Xilinx Spartan-3 Libraries Guide for HDL Designs, ISE 10.1.

[18] Xilinx (April 2008). Spartan-3E FPGA Family: Complete Data Sheet.

[19] Balanced Line. http://www.rane.com/par-b.html#balanced_line.

[20] BEAST - Diffusion: theories and practices, with particular reference to the BEAST system. http://cec.concordia.ca/econtact/Diffusion/Beast.htm.

[21] BEAST - Electroacoustic Music Studios. http://www.beast.bham.ac.uk/.

[22] BEAST - Meet BEAST. http://www.beast.bham.ac.uk/about/meet.shtml.

[23] Digidesign 192 Digital I/O. http://www.digidesign.com/index.cfm?itemid=4891&langid=100.

[24] MAX/MSP - A brief history of MAX. http://freesoftware.ircam.fr/article.php3?id_article=5.

[25] Pro Tools Family. http://www.digidesign.com/index.cfm?navid=349&langid=100&itemid=33116.

[26] SARC - Sonic Arts Research Center. http://143.117.78.181/main.php?page=soniclab.

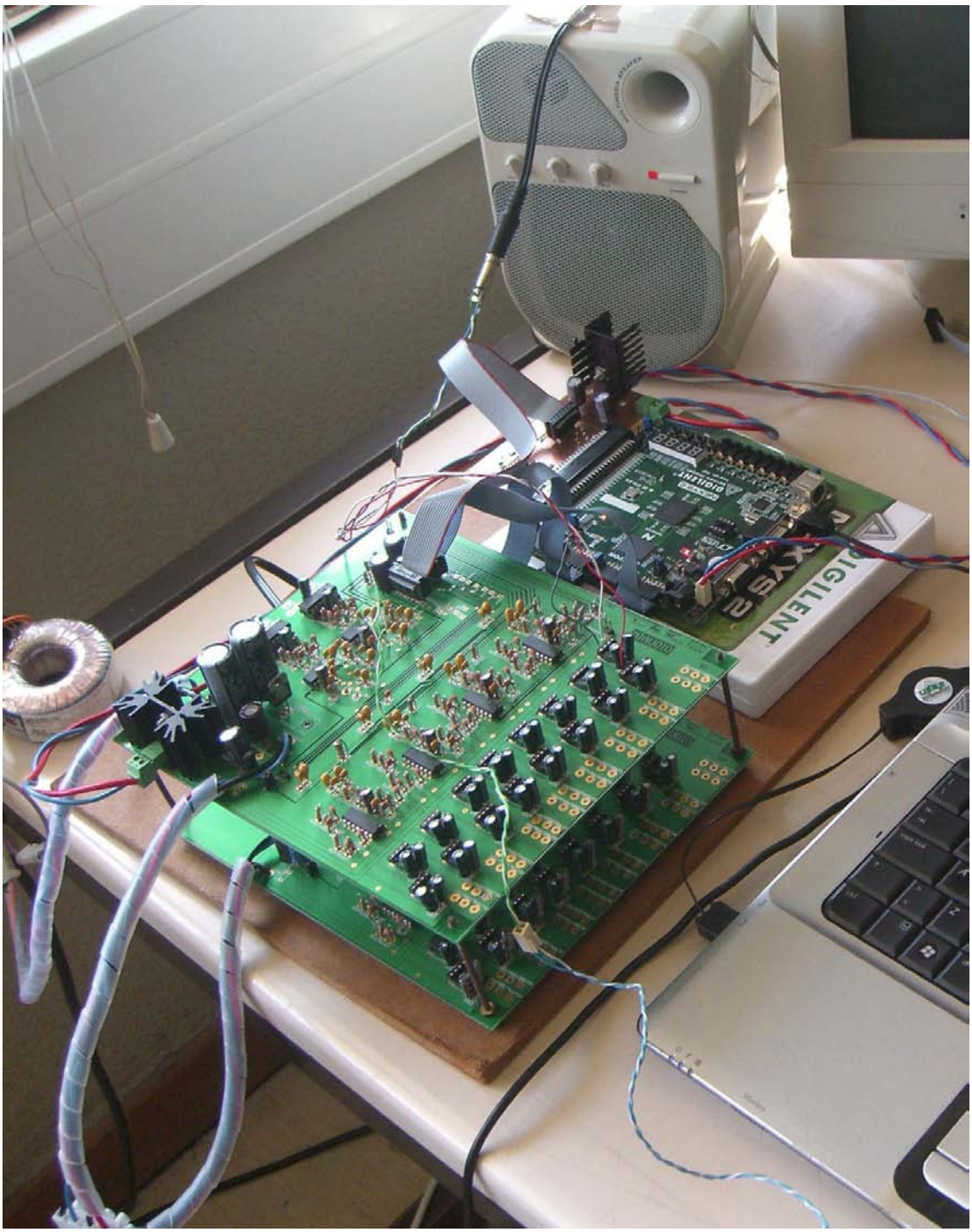
[27] SuperCollider. http://www.audiosynth.com/scfaq.html.

[28] SuperCollider - real-time audio synthesis and algorithmic composition. http://supercollider.sourceforge.net/.

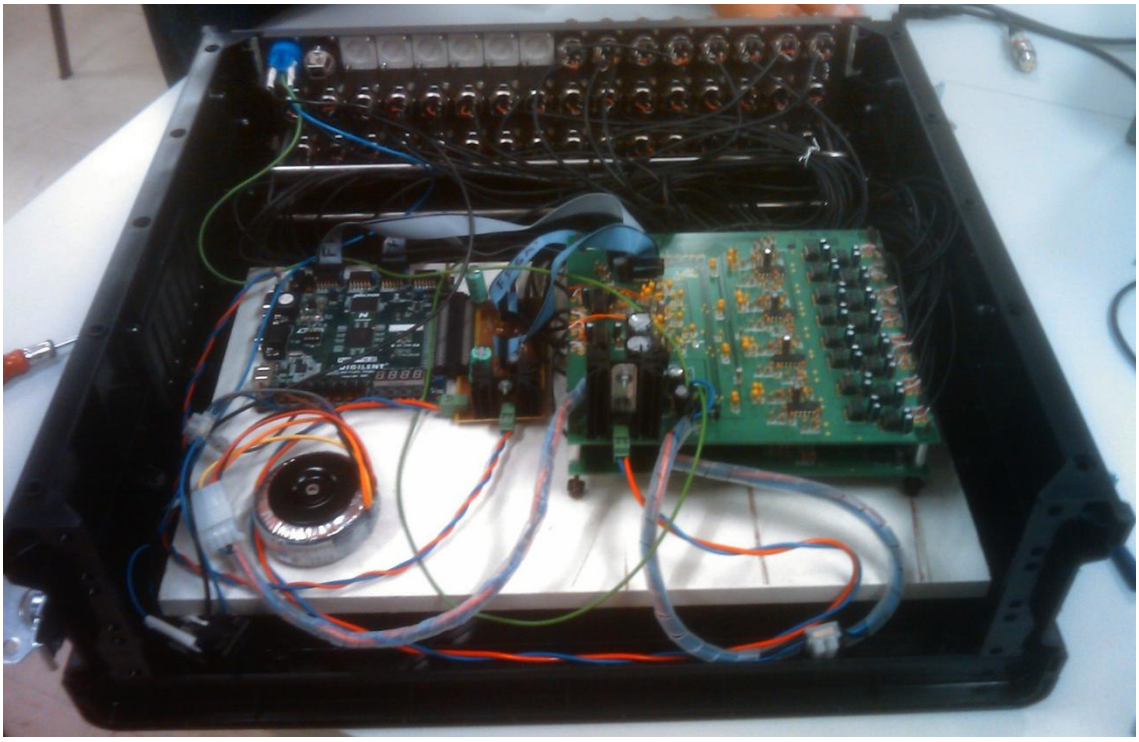
[29] XLR Standard. http://www.rane.com/par-c.html#XLR.

Anexos

Anexo A – Fotografia MIAUDIO (I)



Anexo B – Fotografia MIAUDIO (II)



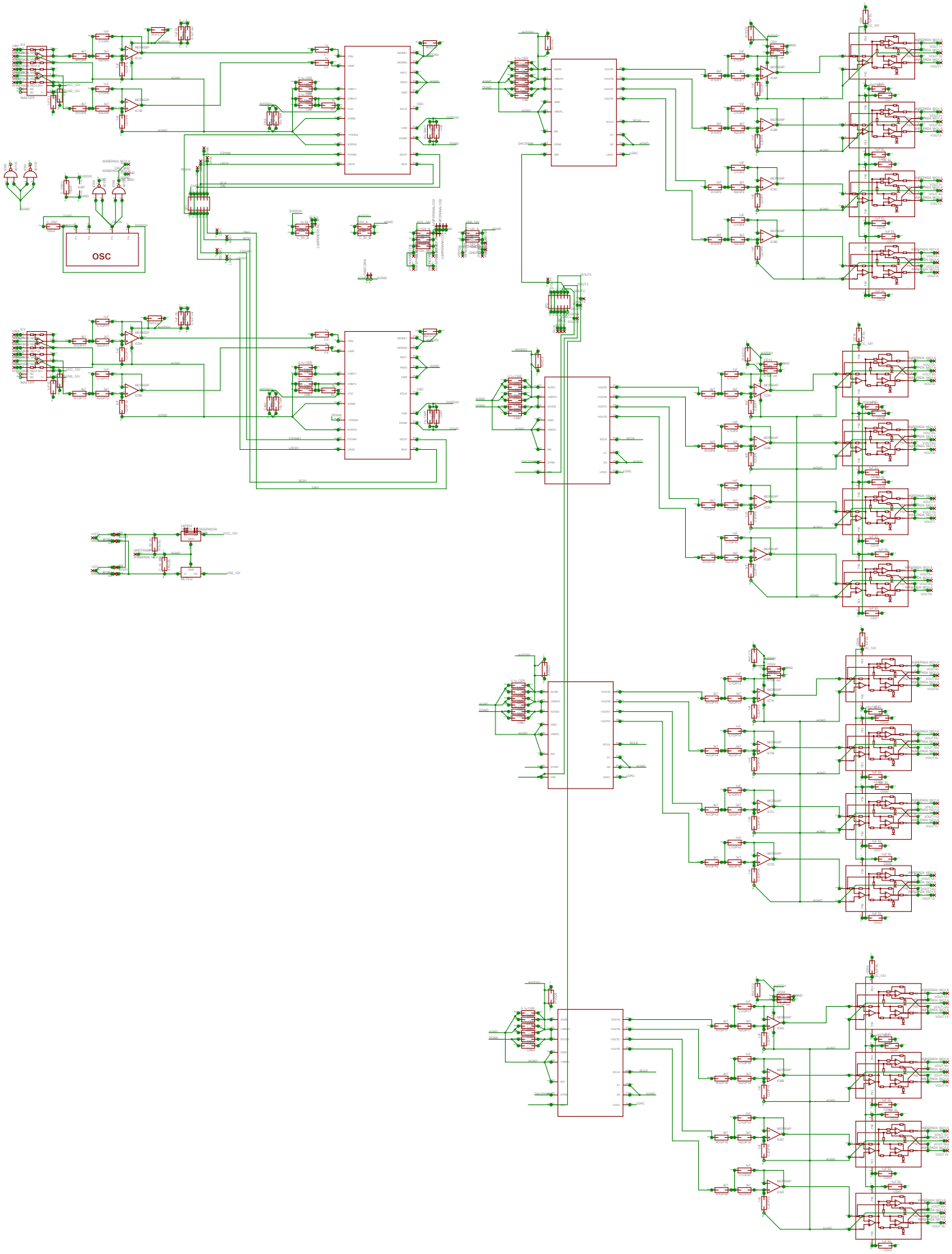
Anexo C – Custo do Sistema

COMPONENTES	PREÇO	QUANTIDADE	SUB-TOTAL		Descrição
DRV134	1,30 €	32	41,60 €		Buffer de Saída
INA2137	1,13 €	4	4,52 €		Buffer de Entrada
PCM1802	2,44 €	4	9,76 €		Conversor Analógico-Digital
DAC8534	5,85 €	8	46,80 €		Conversor Digital-Analógico
MCP602	0,69 €	4	2,76 €		Amplificadores Operacionais - DUAL
MCP604	2,10 €	8	16,80 €		Amplificadores Operacionais - QUAD
LM7805	0,39 €	1	0,39 €		Regulador +5V
LM7812	0,21 €	1	0,21 €		Regulador +12V
MC7912	0,45 €	1	0,45 €		Regulador -12V
HEATSINK	2,76 €	2	5,52 €		Dissipador
NEXYS2	66,18 €	1	66,18 €		Placa Digilent com FPGA
MAIN PCBs (2x)	253,00 €	1	253,00 €		Placas de Circuito Impresso Principais
AUX PCB	30,00 €	1	30,00 €		Placa de Circuito Impresso Auxiliar
		TOTAL	**477,99 €**		

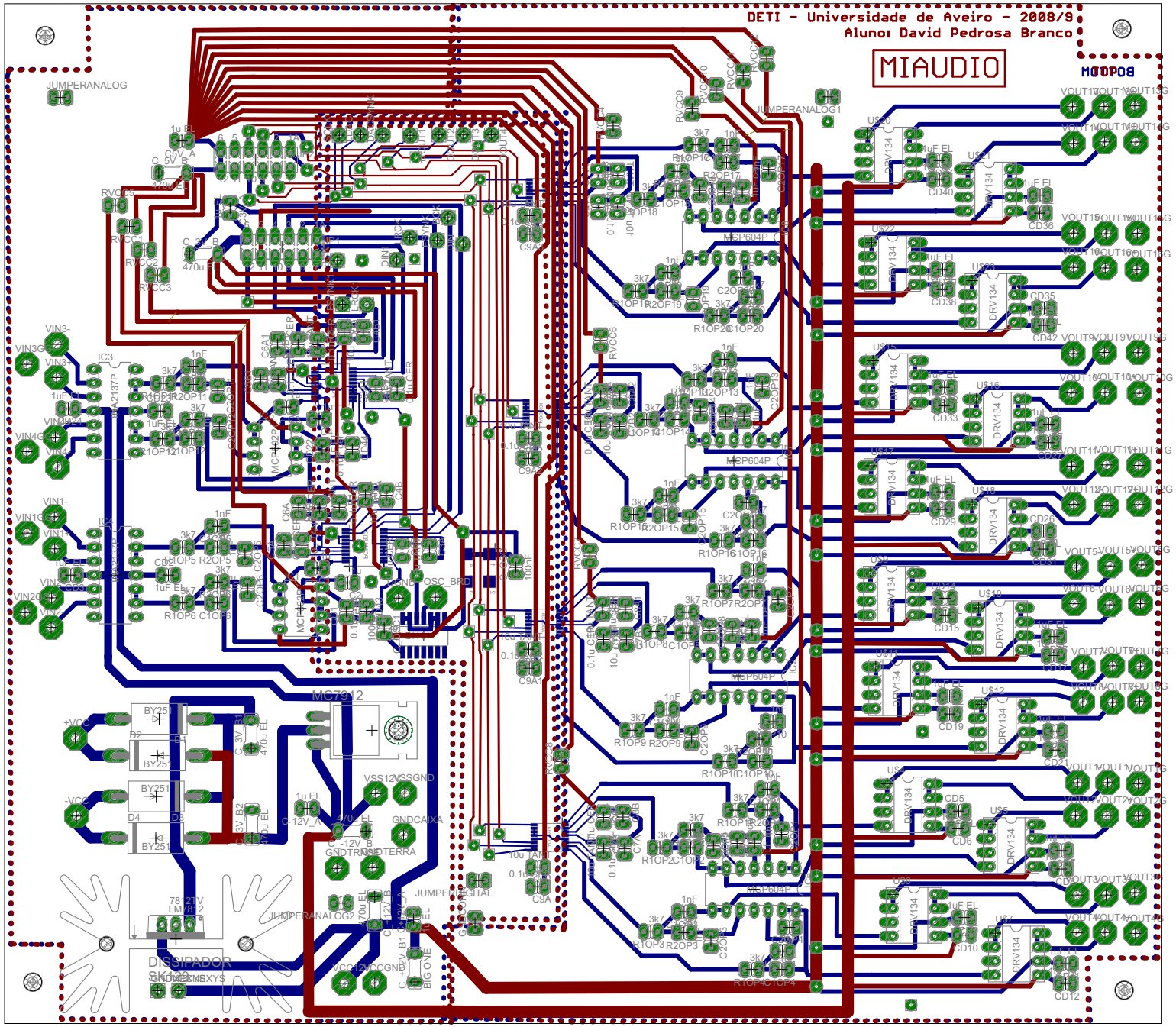
Anexo D – Taxa de Utilização da Lógica da FPGA

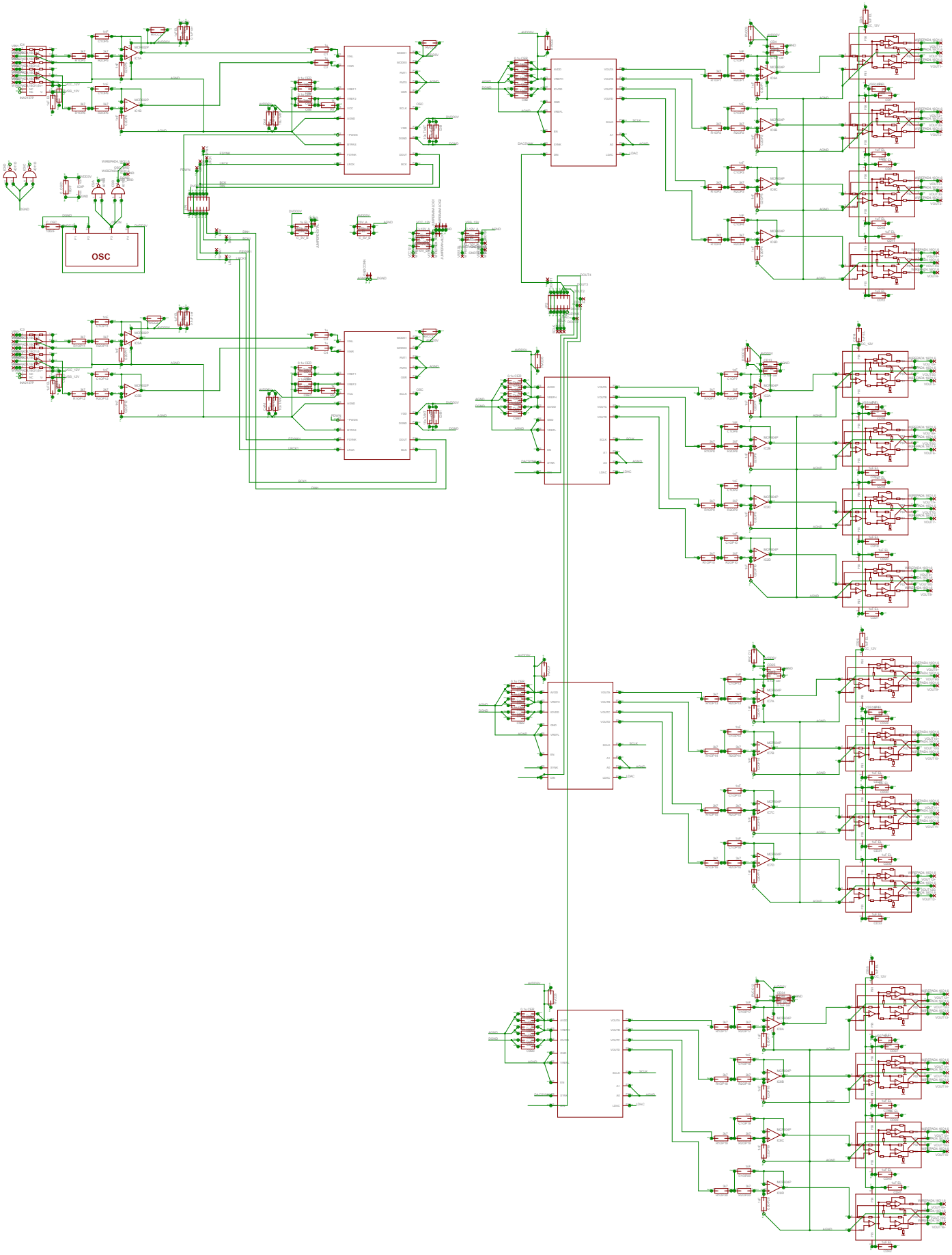
	UTILIZADOS	DISPONÍVEIS	PERCENTAGEM
SLICE FLIP FLOPS	4577	9312	49%
LOOK-UP TABLES	2424	9312	26%
INPUT/OUTPUT BLOCKS	113	232	48%
RAM16	4	20	20%
BUFGMUX	5	24	20%
DCM	1	4	25%
MULT18X18SIO	16	20	80%

Anexo E – Placas de Circuito Impresso

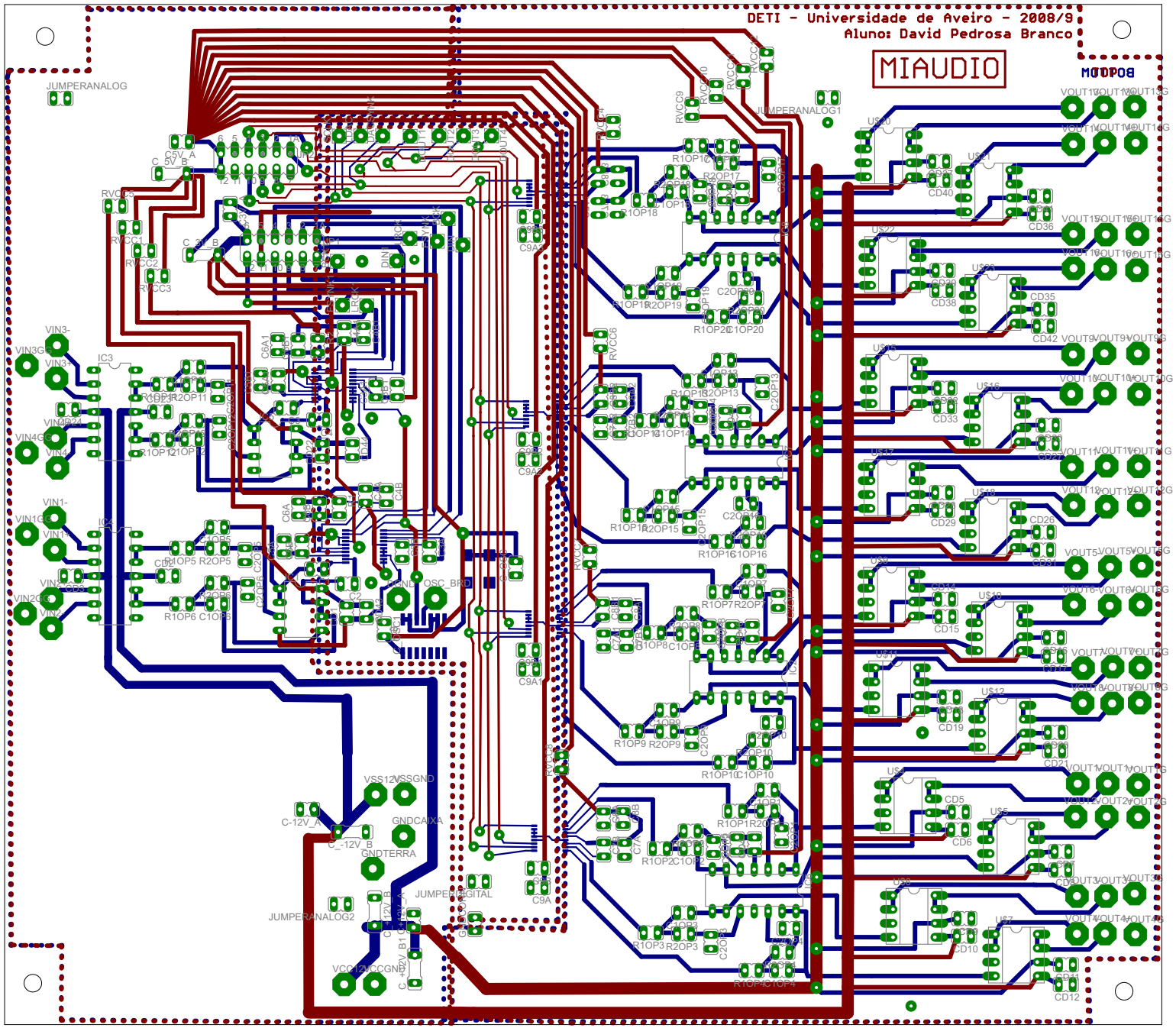

OSC

LAYOUT - PLACA DE CIRCUITO IMPRESSO PRINCIPAL 1/2

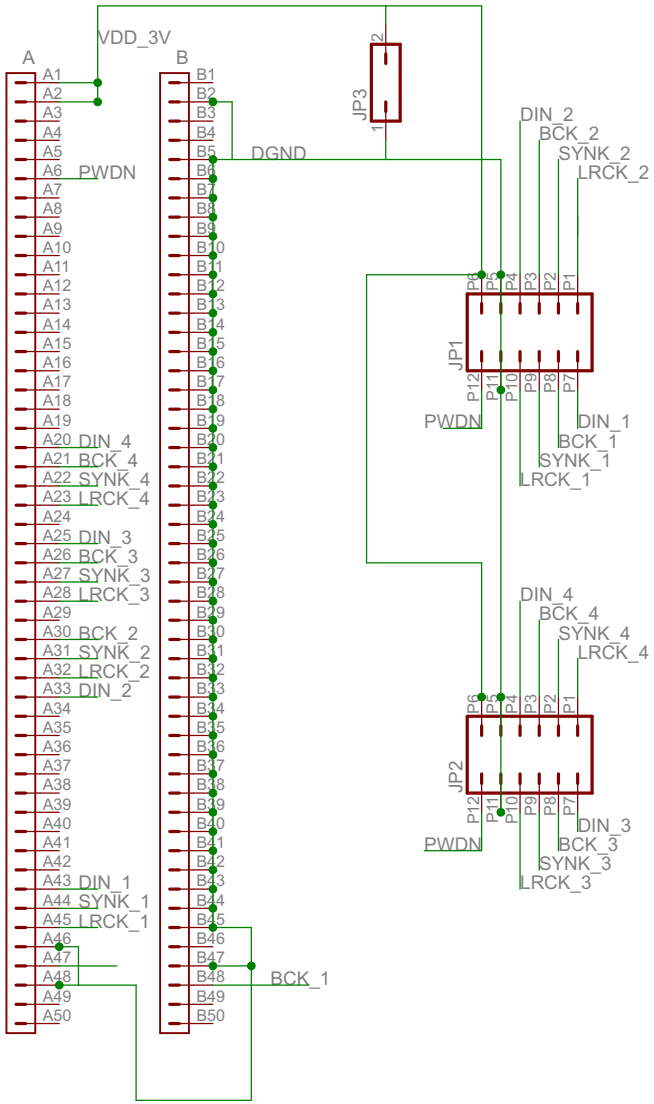


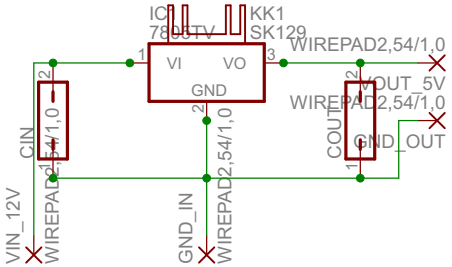

OSC

LAYOUT - PLACA DE CIRCUITO IMPRESSO PRINCIPAL 2/2

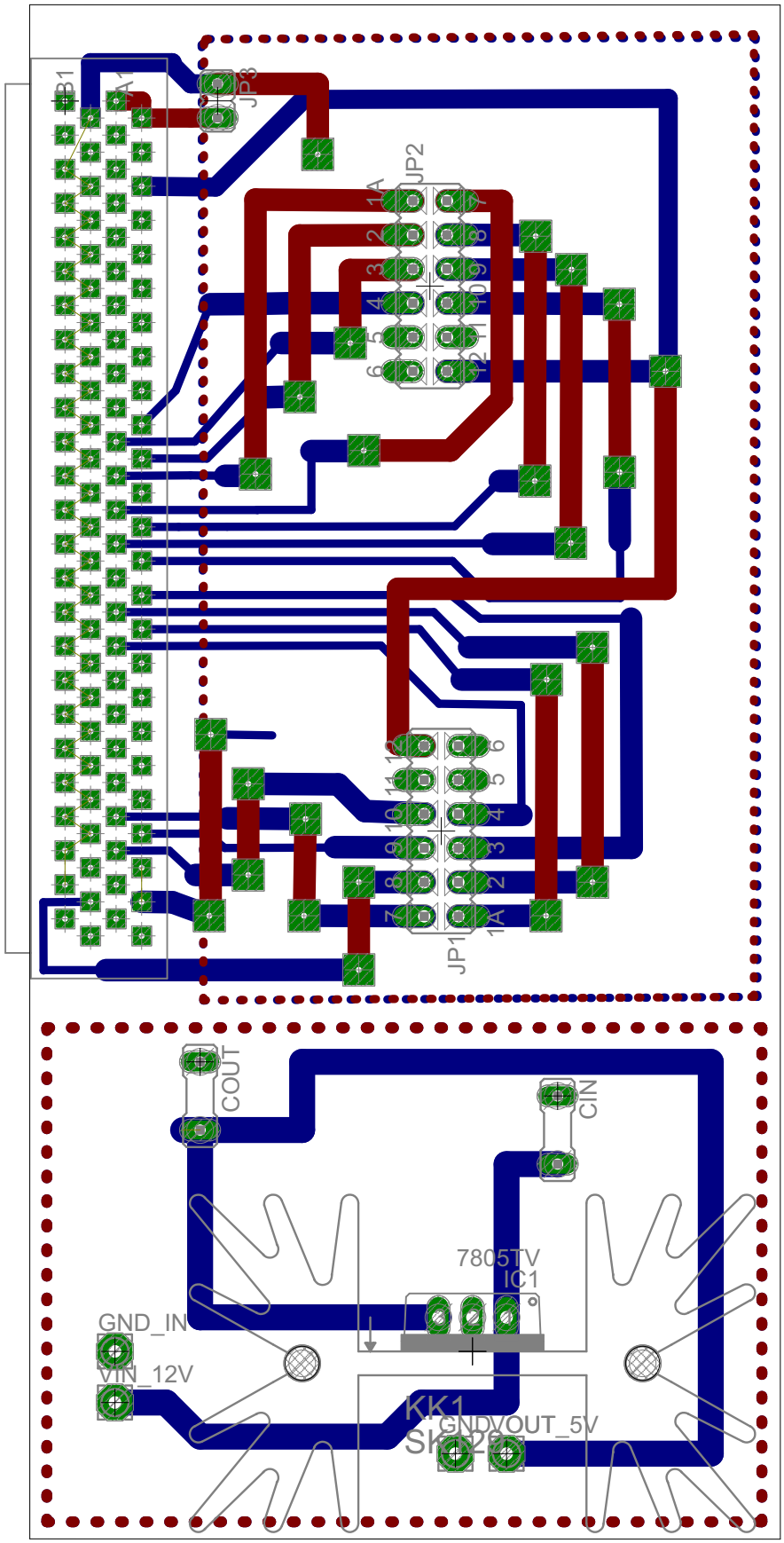


ESQUEMÁTICO - PLACA DE CIRCUITO IMPRESSO AUXILIAR

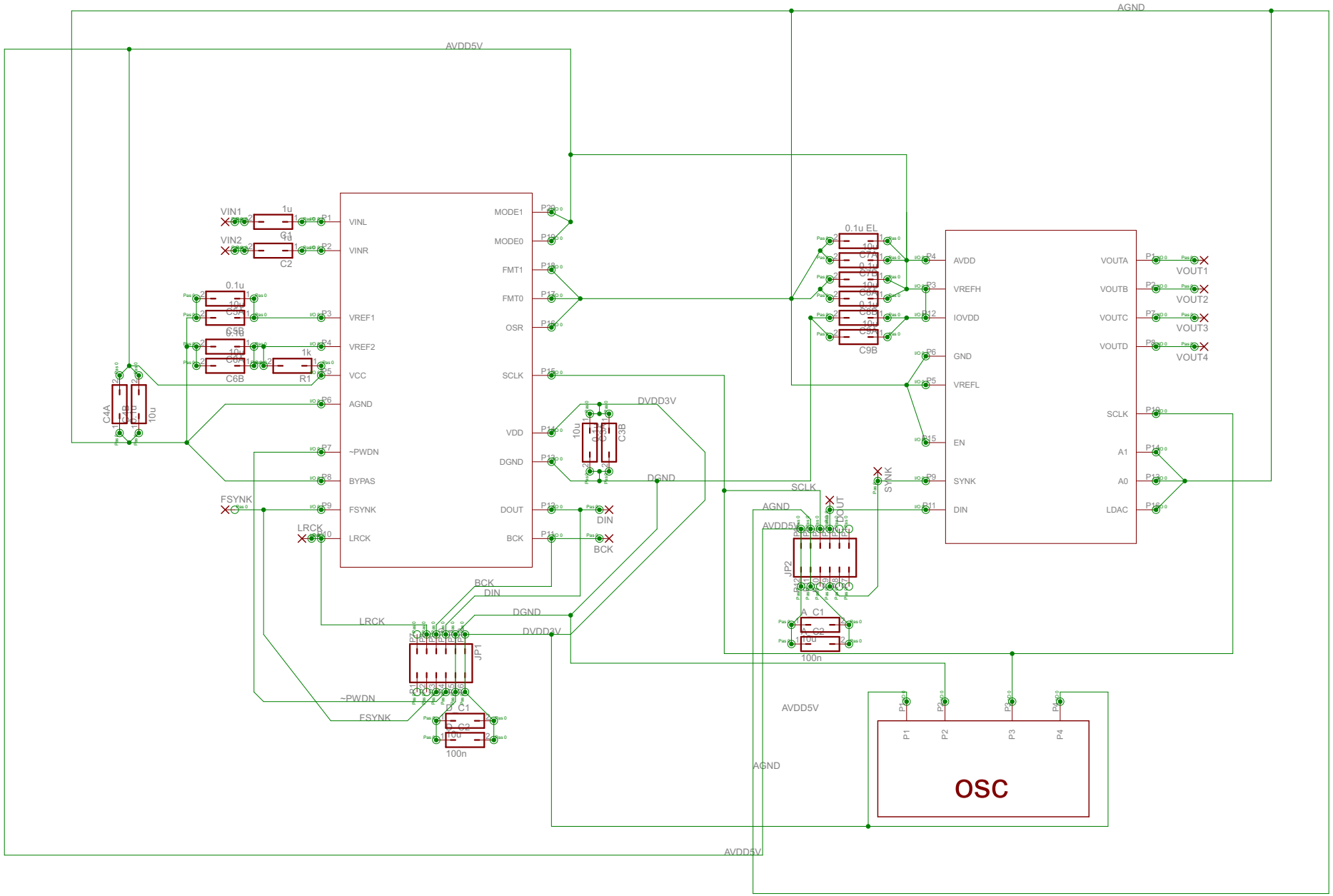




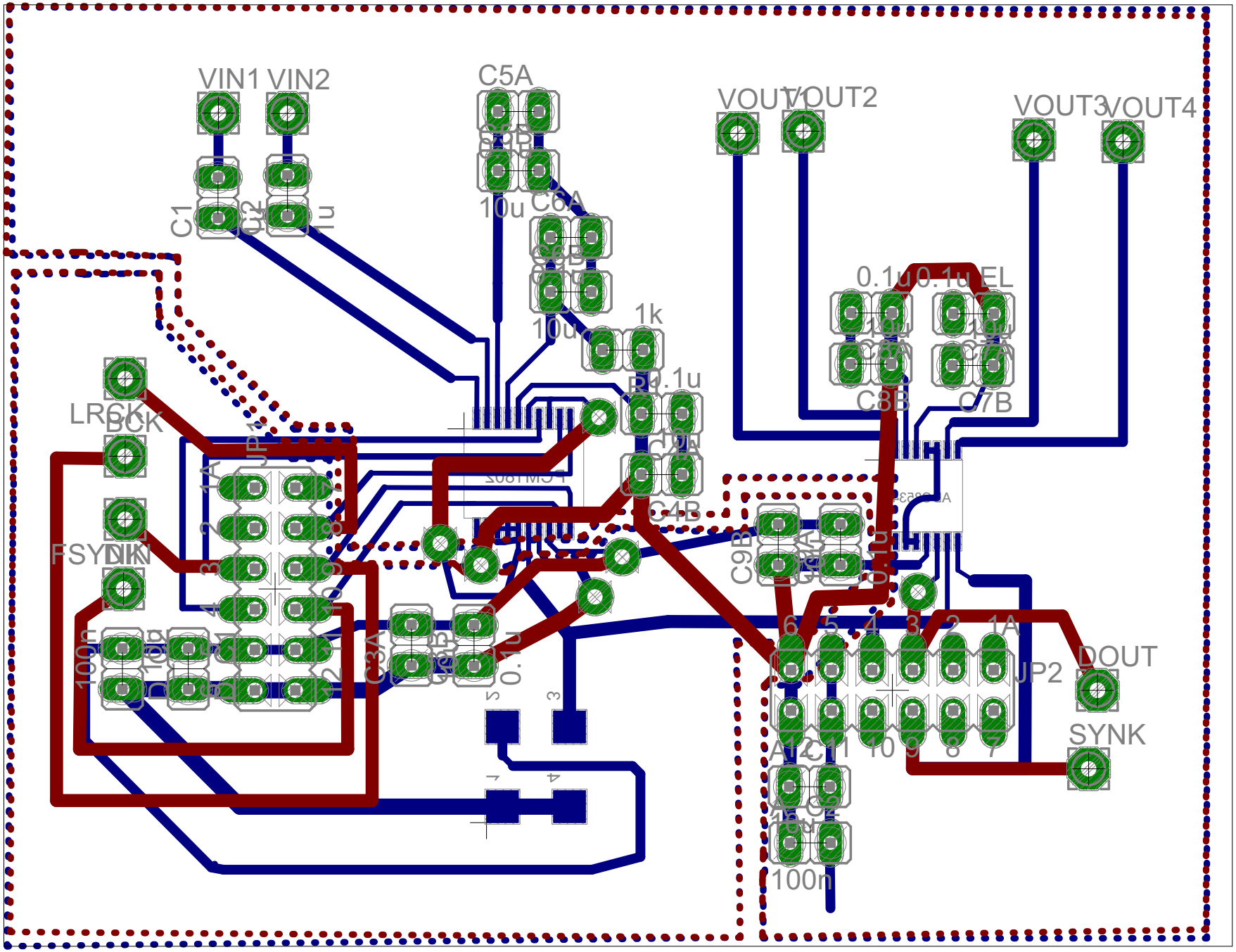
LAYOUT - PLACA DE CIRCUITO IMPRESSO AUXILIAR



ESQUEMÁTICO - PLACA DE CIRCUITO IMPRESSO DE TESTES



LAYOUT - PLACA DE CIRCUITO IMPRESSO DE TESTE



28-10-2009 11:58:24 f=3.00 C:\Users\Ocnarb\Documents\eagle\MIAUDIO\TESTE_BOARD.brd